Foundations and Trends® in Machine Learning
Vol. 8, No. 5-6 (2015) 359–492

DOI: 10.1561/2200000049

now
the essence of knowledge

Bayesian Reinforcement Learning: A Survey

Mohammad Ghavamzadeh
Adobe Research & INRIA
mohammad.ghavamzadeh@inria.fr

Shie Mannor
Technion
shie@ee.technion.ac.il

Joelle Pineau
McGill University
jpineau@cs.mcgill.ca

Aviv Tamar
University of California, Berkeley
avivt@berkeley.edu

Contents

Abstract

Bayesian methods for machine learning have been widely investigated, yielding principled methods for incorporating prior information into inference algorithms. In this survey, we provide an in-depth review of the role of Bayesian methods for the reinforcement learning (RL) paradigm. The major incentives for incorporating Bayesian reasoning in RL are: **1)** it provides an elegant approach to action-selection (exploration/exploitation) as a function of the uncertainty in learning; and **2)** it provides a machinery to incorporate prior knowledge into the algorithms. We first discuss models and methods for Bayesian inference in the simple single-step Bandit model. We then review the extensive recent literature on Bayesian methods for model-based RL, where prior information can be expressed on the parameters of the Markov model. We also present Bayesian methods for model-free RL, where priors are expressed over the value function or policy class. The objective of the paper is to provide a comprehensive survey on Bayesian RL algorithms and their theoretical and empirical properties.

M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. *Bayesian Reinforcement Learning: A Survey*. Foundations and Trends® in Machine Learning, vol. 8, no. 5-6, pp. 359–492, 2015.
DOI: 10.1561/2200000049.

1

Introduction

A large number of problems in science and engineering, from robotics to game playing, tutoring systems, resource management, financial portfolio management, medical treatment design and beyond, can be characterized as sequential decision-making under uncertainty. Many interesting sequential decision-making tasks can be formulated as reinforcement learning (RL) problems [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998]. In an RL problem, an agent interacts with a dynamic, stochastic, and incompletely known environment, with the goal of finding an action-selection strategy, or *policy*, that optimizes some long-term performance measure.

One of the key features of RL is the focus on learning a control policy to optimize the choice of actions over several time steps. This is usually learned from sequences of data. In contrast to supervised learning methods that deal with independently and identically distributed (i.i.d.) samples from the domain, the RL agent learns from the samples that are collected from the trajectories generated by its sequential interaction with the system. Another important aspect is the effect of the agent's policy on the data collection; different policies naturally yield different distributions of sampled trajectories, and thus, impact-

ing what can be learned from the data.

Traditionally, RL algorithms have been categorized as being either *model-based* or *model-free*. In the former category, the agent uses the collected data to first build a model of the domain's dynamics and then uses this model to optimize its policy. In the latter case, the agent directly learns an optimal (or good) action-selection strategy from the collected data. There is some evidence that the first method provides better results with less data [Atkeson and Santamaria, 1997], and the second method may be more efficient in cases where the solution space (e.g., policy space) exhibits more regularity than the underlying dynamics, though there is some disagreement about this,.

A major challenge in RL is in identifying good data collection strategies, that effectively balance between the need to explore the space of all possible policies, and the desire to focus data collection towards trajectories that yield better outcome (e.g., greater chance of reaching a goal, or minimizing a cost function). This is known as the *exploration-exploitation* tradeoff. This challenge arises in both model-based and model-free RL algorithms.

Bayesian reinforcement learning (BRL) is an approach to RL that leverages methods from Bayesian inference to incorporate information into the learning process. It assumes that the designer of the system can express prior information about the problem in a probabilistic distribution, and that new information can be incorporated using standard rules of Bayesian inference. The information can be encoded and updated using a parametric representation of the system dynamics, in the case of model-based RL, or of the solution space, in the case of model-free RL.

A major advantage of the BRL approach is that it provides a principled way to tackle the exploration-exploitation problem. Indeed, the Bayesian posterior naturally captures the full state of knowledge, subject to the chosen parametric representation, and thus, the agent can select actions that maximize the expected gain with respect to this information state.

Another major advantage of BRL is that it implicitly facilitates regularization. By assuming a prior on the value function, the parameters

defining a policy, or the model parameters, we avoid the trap of letting a few data points steer us away from the true parameters. On the other hand, having a prior precludes overly rapid convergence. The role of the prior is therefore to soften the effect of sampling a finite dataset, effectively leading to regularization. We note that regularization in RL has been addressed for the value function [Farahmand et al., 2008b] and for policies [Farahmand et al., 2008a]. A major issue with these regularization schemes is that it is not clear how to select the regularization coefficient. Moreover, it is not clear why an optimal value function (or a policy) should belong to some pre-defined set.

Yet another advantage of adopting a Bayesian view in RL is the principled Bayesian approach for handling parameter uncertainty. Current frequentist approaches for dealing with modelling errors in sequential decision making are either very conservative, or computationally infeasible [Nilim and El Ghaoui, 2005]. By explicitly modelling the distribution over unknown system parameters, Bayesian methods offer a promising approach for solving this difficult problem.

Of course, several challenges arise in applying Bayesian methods to the RL paradigm. First, there is the challenge of selecting the correct representation for expressing prior information in any given domain. Second, defining the decision-making process over the information state is typically computationally more demanding than directly considering the natural state representation. Nonetheless, a large array of models and algorithms have been proposed for the BRL framework, leveraging a variety of structural assumptions and approximations to provide feasible solutions.

The main objective of this paper is to provide a comprehensive survey on BRL algorithms and their theoretical and empirical properties. In Chapter 2, we provide a review of the main mathematical concepts and techniques used throughout this paper. Chapter 3 surveys the Bayesian learning methods for the case of single-step decision-making, using the *bandit* framework. This section serves both as an exposition of the potential of BRL in a simpler setting that is well understood, but is also of independent interest, as bandits have widespread applications. The main results presented here are of a theoretical nature, outlin-

ing known performance bounds for the regret minimization criteria. Chapter 4 reviews existing methods for *model-based* BRL, where the posterior is expressed over parameters of the system dynamics model. Chapter 5 focuses on BRL methods that do not explicitly learn a model of the system, but rather the posterior is expressed over the solution space. Chapter 6 focuses on a particular advantage of BRL in dealing with risk due to parameter-uncertainty, and surveys several approaches for incorporating such risk into the decision-making process. Finally, Chapter 7 discusses various extensions of BRL for special classes of problems (PAC-Bayes model selection, inverse RL, multi-agent RL, and multi-task RL). Figure 1.1 outlines the various BRL approaches covered throughout the paper.

An Example Domain

We present an illustrative domain suitable to be solved using the BRL techniques surveyed in this paper. This running example will be used throughout the paper to elucidate the difference between the various BRL approaches and to clarify various BRL concepts.

Example 1.1 (The Online Shop)**.** In the online shop domain, a retailer aims to maximize profit by sequentially suggesting products to online shopping customers. Formally, the domain is characterized by the following model:

- A set of possible customer states, $\mathcal{X}$. States can represent intrinsic features of the customer such as gender and age, but also dynamic quantities such as the items in his shopping cart, or his willingness to shop;
- A set of possible product suggestions and advertisements, $\mathcal{A}$;
- A probability kernel, P, defined below.

An ***episode*** in the online shop domain begins at time $t = 0$, when a customer with features $x_0 \in \mathcal{X}$ enters the online shop. Then, a sequential interaction between the customer and the online shop begins, where at each step $t = 0, 1, 2, \ldots$, an advertisement $a_t \in \mathcal{A}$ is shown

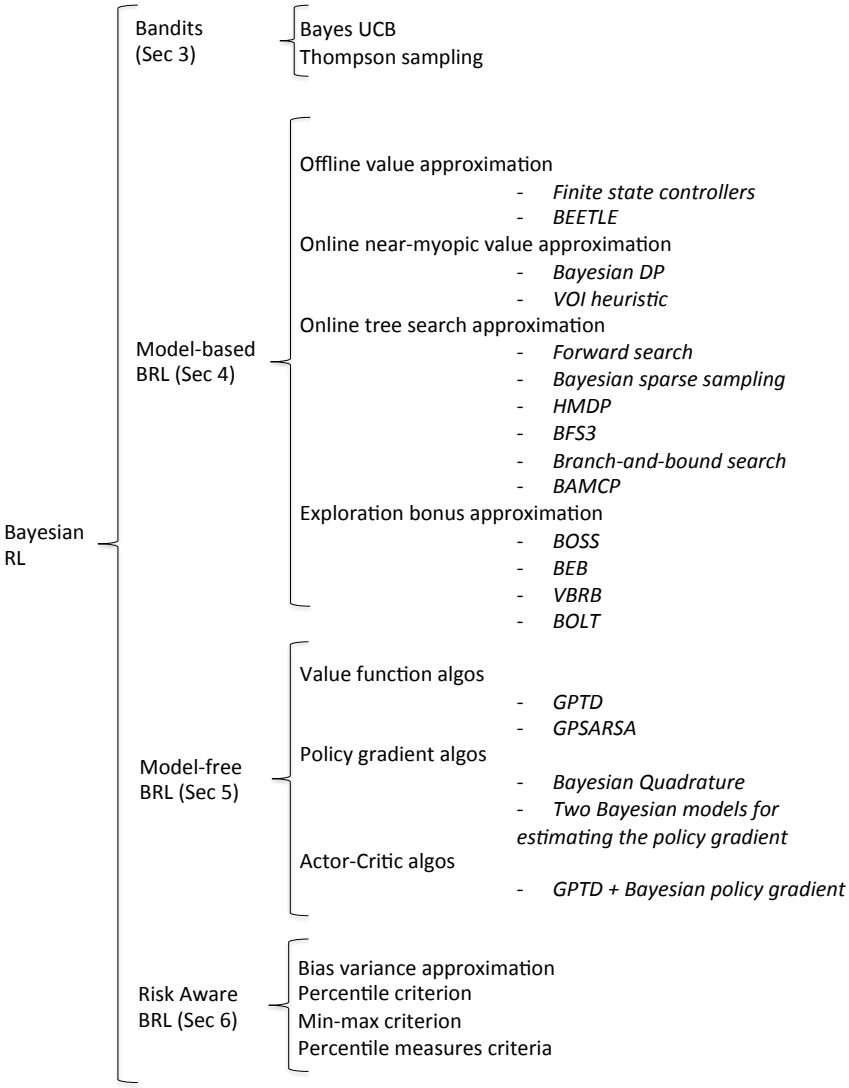

Figure 1.1: Overview of the Bayesian RL approaches covered in this survey.

to the customer, and following that the customer makes a decision to either (i) add a product to his shopping cart; (ii) not buy the product, but continue to shop; (iii) stop shopping and check out. Following the customers decision, his state changes to x_{t+1} (reflecting the change in the shopping cart, willingness to continue shopping, etc.). We assume that this change is captured by a probability kernel $P(x_{t+1}|\, x_t, a_t)$.

When the customer decides to check out, the episode ends, and a profit is obtained according to the items he had added to his cart. The goal is to find a product suggestion policy, $x \to a \in \mathcal{A}$, that maximizes the expected total profit.

When the probabilities of customer responses P are known in advance, calculating an optimal policy for the online shop domain is basically a *planning* problem, which may be solved using traditional methods for resource allocation [Powell, 2011]. A more challenging, but realistic, scenario is when P is not completely known beforehand, but has to be *learned* while interacting with customers. The BRL framework employs *Bayesian* methods for learning P, and for learning an optimal product suggestion policy.

There are several advantages for choosing a Bayesian approach for the online shop domain. First, it is likely that some prior knowledge about P is available. For example, once a customer adds a product of a particular brand to his cart, it is likely that he prefers additional products of the same brand over those of a different one. Taking into account such knowledge is natural in the Bayesian method, by virtue of the *prior* distribution over P. As we shall see, the Bayesian approach also naturally extends to more general forms of *structure* in the problem.

A second advantage concerns what is known as the *exploitation–exploration* dilemma: should the decision-maker display only the most profitable product suggestions according to his current knowledge about P, or rather take exploratory actions that may turn out to be less profitable, but provide useful information for future decisions? The Bayesian method offers a principled approach to dealing with this difficult problem by explicitly quantifying the value of exploration, made possible by maintaining a *distribution* over P.

The various parameter configurations in the online shop domain lead to the different learning problems surveyed in this paper. In particular:

- For a single-step interaction, i.e., when the episode terminates after a single product suggestion, the problem is captured by the multi-armed bandit model of Chapter 3.
- For small-scale problems, i.e., a small number of products and customer types, P may be learnt explicitly. This is the model-based approach of Chapter 4.
- For large problems, a near-optimal policy may be obtained without representing P explicitly. This is the model-free approach of Chapter 5.
- When the customer state is not fully observed by the decision-maker, we require models that incorporate partial observability; see §2.3 and §4.9.

Throughout the paper, we revisit the online shop domain, and specify explicit configurations that are relevant to the surveyed methods.

2

Technical Background

In this section we provide a brief overview of the main concepts and introduce the notations used throughout the paper. We begin with a quick review of the primary mathematical tools and algorithms leveraged in the latter sections to define BRL models and algorithms.

2.1 Multi-Armed Bandits

As was previously mentioned, a key challenge in sequential decision-making under uncertainty is the exploration/exploitation dilemma: the tradeoff between either taking actions that are most rewarding according to the current state of knowledge, or taking exploratory actions, which may be less immediately rewarding, but may lead to better informed decisions in the future.

The *stochastic multi-armed bandit* (MAB) problem is perhaps the simplest model of the exploration/exploitation tradeoff. Originally formulated as the problem of a gambler choosing between different slot machines in a casino ('one armed bandits'), the stochastic MAB model features a decision-maker that sequentially chooses actions $a_t \in \mathcal{A}$, and observes random outcomes $Y_t(a_t) \in \mathcal{Y}$ at discrete time steps

$t = 1, 2, 3, \ldots$. A known function $r : \mathcal{Y} \to \mathbb{R}$ associates the random outcome to a reward, which the agent seeks to maximize. The outcomes $\{Y_t(a)\}$ are drawn i.i.d. over time from an unknown probability distribution $P(\cdot|a) \in \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ denotes the set of probability distributions on (Borel) subsets of $\mathcal{Y}$ (see Bubeck and Cesa-Bianchi [2012] for reference).

Model 1 (Stochastic K-Armed Bandit) Define a K-MAB to be a tuple $\langle \mathcal{A}, \mathcal{Y}, P, r \rangle$ where
• $\mathcal{A}$ is the set of actions (arms), and $|\mathcal{A}| = K$,
• $\mathcal{Y}$ is the set of possible outcomes,
• $P(\cdot|a) \in \mathcal{P}(\mathcal{Y})$ is the outcome probability, conditioned on action $a \in \mathcal{A}$ being taken,
• $r(Y) \in \mathbb{R}$ represents the reward obtained when outcome $Y \in \mathcal{Y}$ is observed.

A rule that prescribes to the agent which actions to select, or *policy*, is defined as a mapping from past observations to a distribution over the set of actions. Since the probability distributions are initially unknown, the decision-maker faces a tradeoff between exploitation, i.e., selecting the arm she believes is optimal, or making exploratory actions to gather more information about the true probabilities. Formally, this tradeoff is captured by the notion of *regret*:

Definition 2.1 (Regret)**.** Let $a^* \in \arg\max_{a \in \mathcal{A}} \mathbf{E}_{y \sim P(\cdot|a)}\big[r(y)\big]$ denote the optimal arm. The T-period regret of the sequence of actions $a_1, \ldots, a_T$ is the random variable

$$\mathbf{Regret}(T) = \sum_{t=1}^{T} \Big[r\big(Y_t(a^*)\big) - r\big(Y_t(a_t)\big) \Big].$$

Typically, MAB algorithms focus on the *expected regret*, $\mathbf{E}\big[\mathbf{Regret}(T)\big]$, and provide policies that are guaranteed to keep it small in some sense.

As an illustration of the MAB model, let us revisit the online shop example.

Example 2.1 (Online shop – bandit setting)**.** Recall the online shop domain of Example 1.1. In the MAB setting, there is no state information

about the customers, i.e., $\mathcal{X} = \emptyset$. In addition, each interaction lasts a single time step, after which the customer checks out, and a profit is obtained according to his purchasing decision. The regret minimization problem corresponds to determining which sequence of advertisements $a_1, \ldots, a_T$ to show to a stream of T incoming customers, such that the total revenue is close to that which would have been obtained with the optimal advertisement stream.

In some cases, the decision-maker may have access to some additional information that is important for making decisions, but is not captured in the MAB model. For example, in the online shop domain of Example 2.1, it is likely that some information about the customer, such as his age or origin, is available. The *contextual bandit* model (a.k.a. associative bandits, or bandits with side information) is an extension of the MAB model that takes such information into account.

The decision-making process in the contextual bandit model is similar to the MAB case. However, at each time step t, the decision maker first observes a context $s_t \in \mathcal{S}$, drawn i.i.d. over time from a probability distribution $P_S(\cdot) \in \mathcal{P}(\mathcal{S})$, where $\mathcal{P}(\mathcal{S})$ denotes the set of probability distributions on (Borel) subsets of $\mathcal{S}$. The decision-maker then chooses an action $a_t \in \mathcal{A}$ and observes a random outcome $Y_t(a_t, s_t) \in \mathcal{Y}$, which now depends both on the context and the action. The outcomes $\{Y_t(a)\}$ are drawn i.i.d. over time from an unknown probability distribution $P(\cdot|a, s) \in \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ denotes the set of probability distributions on (Borel) subsets of $\mathcal{Y}$.

Model 2 (Contextual Bandit) Define a contextual bandit to be a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Y}, P_S, P, r \rangle$ where
- $\mathcal{S}$ is the set of contexts,
- $\mathcal{A}$ is the set of actions (arms),
- $\mathcal{Y}$ is the set of possible outcomes,
- $P_S(\cdot) \in \mathcal{P}(\mathcal{S})$ is the context probability,
- $P(\cdot|a, s) \in \mathcal{P}(\mathcal{Y})$ is the outcome probability, conditioned on action $a \in \mathcal{A}$ being taken when the context is $s \in \mathcal{S}$,
- $r(Y) \in \mathbb{R}$ represents the reward obtained when outcome $Y \in \mathcal{Y}$ is observed.

The Markov decision process model, as presented in the following section, may be seen as an extension of the contextual bandit model to a *sequential* decision-making model, in which the context is no longer i.i.d., but may change over time according to the selected actions.

2.2 Markov Decision Processes

The Markov Decision Process (MDP) is a framework for sequential decision-making in Markovian dynamical systems [Bellman, 1957, Puterman, 1994]. It can be seen as an extension of the MAB framework by adding the notion of a system *state*, that may dynamically change according to the performed actions and affects the outcomes of the system.

Model 3 (Markov Decision Process) Define an MDP $\mathcal{M}$ to be a tuple $\langle \mathcal{S}, \mathcal{A}, P, P_0, q \rangle$ where
• $\mathcal{S}$ is the set of states,
• $\mathcal{A}$ is the set of actions,
• $P(\cdot|s,a) \in \mathcal{P}(\mathcal{S})$ is the probability distribution over next states, conditioned on action a being taken in state s,
• $P_0 \in \mathcal{P}(\mathcal{S})$ is the probability distribution according to which the initial state is selected,
• $R(s,a) \sim q(\cdot|s,a) \in \mathcal{P}(\mathbb{R})$ is a random variable representing the reward obtained when action a is taken in state s.

Let $\mathcal{P}(\mathcal{S})$, $\mathcal{P}(\mathcal{A})$, and $\mathcal{P}(\mathbb{R})$ be the set of probability distributions on (Borel) subsets of $\mathcal{S}$, $\mathcal{A}$, and $\mathbb{R}$ respectively.[1] We assume that P, P_0 and q are stationary. Throughout the paper, we use upper-case and lower-case letters to refer to random variables and the values taken by random variables, respectively. For example, $R(s,a)$ is the random variable of the immediate reward, and $r(s,a)$ is one possible realization of this random variable. We denote the expected value of $R(s,a)$, as $\bar{r}(s,a) = \int r \; q(dr|s,a)$.

Assumption A1 (MDP Regularity) We assume that the random immediate rewards are bounded by $R_{\max}$ and the expected immediate

[1] $\mathbb{R}$ is the set of real numbers.

rewards are bounded by $\bar{R}_{\max}$. Note that $\bar{R}_{\max} \leq R_{\max}$.

A rule according to which the agent selects its actions at each possible state, or *policy*, is defined as a mapping from past observations to a distribution over the set of actions. A policy is called *Markov* if the distribution depends only on the last state of the observation sequence. A policy is called *stationary* if it does not change over time. A stationary Markov policy $\mu(\cdot|s) \in \mathcal{P}(\mathcal{A})$ is a probability distribution over the set of actions given a state $s \in \mathcal{S}$. A policy is *deterministic* if the probability distribution concentrates on a single action for all histories. A deterministic policy is identified by a mapping from the set of states to the set of actions, i.e., $\mu : \mathcal{S} \to \mathcal{A}$. In the rest of the paper, we use the term policy to refer to stationary Markov policies.

The MDP controlled by a policy μ induces a Markov chain $\mathcal{M}^\mu$ with reward distribution $q^\mu(\cdot|s) = q(\,\cdot\,|s, \mu(s))$ such that $R^\mu(s) = R\big(s, \mu(s)\big) \in q^\mu(\cdot|s)$, transition kernel $P^\mu(\cdot|s) = P(\,\cdot\,|s, \mu(s))$, and stationary distribution over states π^μ (if it admits one). In a Markov chain $\mathcal{M}^\mu$, for state-action pairs $z = (s, a) \in \mathcal{Z} = \mathcal{S} \times \mathcal{A}$, we define the transition density and the initial (state-action) density as $P^\mu(z'|z) = P(s'|s, a)\mu(a'|s')$ and $P_0^\mu(z_0) = P_0(s_0)\mu(a_0|s_0)$, respectively. We generically use $\xi = \{z_0, z_1, \ldots, z_T\} \in \Xi$, $T \in \{0, 1, \ldots, \infty\}$, to denote a path (or trajectory) generated by this Markov chain. The probability (density) of such a path is given by

$$\Pr(\xi|\mu) = P_0^\mu(z_0) \prod_{t=1}^{T} P^\mu(z_t|z_{t-1}).$$

We define the (possibly discounted, $\gamma \in [0, 1]$) return of a path as a function $\rho : \Xi \to \mathbb{R}$, $\rho(\xi) = \sum_{t=0}^{T} \gamma^t R(z_t)$. For each path ξ, its (discounted) return, $\rho(\xi)$, is a random variable with the expected value $\bar{\rho}(\xi) = \sum_{t=0}^{T} \gamma^t \bar{r}(z_t)$.[2] Here, $\gamma \in [0, 1]$ is a discount factor that determines the exponential devaluation rate of delayed rewards.[3] The

[2] When there is no randomness in the rewards, i.e., $r(s, a) = \bar{r}(s, a)$, then $\rho(\xi) = \bar{\rho}(\xi)$.

[3] When $\gamma = 1$, the policy must be proper, i.e., guaranteed to terminate, see [Puterman, 1994] or [Bertsekas and Tsitsiklis, 1996] for more details.

expected return of a policy μ is defined by

$$\eta(\mu) = \mathbf{E}[\rho(\xi)] = \int_{\Xi} \bar{\rho}(\xi) \Pr(\xi|\mu) d\xi. \tag{2.1}$$

The expectation is over all possible trajectories generated by policy μ and all possible rewards collected in them.

Similarly, for a given policy μ, we can define the (discounted) return of a state s, $D^\mu(s)$, as the sum of (discounted) rewards that the agent encounters when it starts in state s and follows policy μ afterwards

$$D^\mu(s) = \sum_{t=0}^{\infty} \gamma^t R(Z_t) \mid Z_0 = (s, \mu(\cdot|s)), \quad \text{with } S_{t+1} \sim P^\mu(\cdot|S_t). \tag{2.2}$$

The expected value of D^μ is called the value function of policy μ

$$V^\mu(s) = \mathbf{E}[D^\mu(s)] = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(Z_t) \,\middle|\, Z_0 = (s, \mu(\cdot|s)) \right]. \tag{2.3}$$

Closely related to value function is the action-value function of a policy, the total expected (discounted) reward observed by the agent when it starts in state s, takes action a, and then executes policy μ

$$Q^\mu(z) = \mathbf{E}[D^\mu(z)] = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(Z_t) \,\middle|\, Z_0 = z \right],$$

where similarly to $D^\mu(s)$, $D^\mu(z)$ is the sum of (discounted) rewards that the agent encounters when it starts in state s, takes action a, and follows policy μ afterwards. It is easy to see that for any policy μ, the functions V^μ and Q^μ are bounded by $\bar{R}_{\max}/(1-\gamma)$. We may write the value of a state s under a policy μ in terms of its immediate reward and the values of its successor states under μ as

$$V^\mu(s) = R^\mu(s) + \gamma \int_{\mathcal{S}} P^\mu(s'|s) V^\mu(s') ds', \tag{2.4}$$

which is called the *Bellman equation* for V^μ.

Given an MDP, the goal is to find a policy that attains the best possible values, $V^*(s) = \sup_\mu V^\mu(s)$, for all states $s \in \mathcal{S}$. The function V^* is called the *optimal* value function. A policy is optimal (denoted by μ^*) if it attains the optimal values at all the states, i.e., $V^{\mu^*}(s) = V^*(s)$

for all $s \in \mathcal{S}$. In order to characterize optimal policies it is useful to define the optimal action-value function

$$Q^*(z) = \sup_{\mu} Q^{\mu}(z). \tag{2.5}$$

Further, we say that a deterministic policy μ is *greedy* with respect to an action-value function Q, if for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, $\mu(s) \in \arg\max_{a \in \mathcal{A}} Q(s, a)$. Greedy policies are important because any greedy policy with respect to Q^* is optimal. Similar to the value function of a policy (Eq. 2.4), the optimal value function of a state s may be written in terms of the optimal values of its successor states as

$$V^*(s) = \max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \int_{\mathcal{S}} P(s'|s, a) V^*(s') ds' \right], \tag{2.6}$$

which is called the *Bellman optimality equation*. Note that, similar to Eqs. 2.4 and 2.6, we may define the Bellman equation and the Bellman optimality equation for action-value function.

It is important to note that almost all methods for finding the optimal solution of an MDP are based on two *dynamic programming* (DP) algorithms: *value iteration* and *policy iteration*. Value iteration (VI) begins with a value function V_0 and at each iteration i, generates a new value function by applying Eq. 2.6 to the current value function, i.e.,

$$V_i^*(s) = \max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \int_{\mathcal{S}} P(s'|s, a) V_{i-1}^*(s') ds' \right], \qquad \forall s \in \mathcal{S}.$$

Policy iteration (PI) starts with an initial policy. At each iteration, it evaluates the value function of the current policy, this process is referred to as *policy evaluation* (PE), and then performs a *policy improvement* step, in which a new policy is generated as a greedy policy with respect to the value of the current policy. Iterating the policy evaluation – policy improvement process is known to produce a strictly monotonically improving sequence of policies. If the improved policy is the same as the policy improved upon, then we are assured that the optimal policy has been found.

2.3 Partially Observable Markov Decision Processes

The Partially Observable Markov Decision Process (POMDP) is an extension of MDP to the case where the state of the system is not necessarily observable [Astrom, 1965, Smallwood and Sondik, 1973, Kaelbling et al., 1998].

Model 4 (Partially Observable Markov Decision Process) Define a POMDP $\mathcal{M}$ to be a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P, \Omega, P_0, q \rangle$ where
• $\mathcal{S}$ is the set of states,
• $\mathcal{A}$ is the set of actions,
• $\mathcal{O}$ is the set of observations,
• $P(\cdot|s,a) \in \mathcal{P}(\mathcal{S})$ is the probability distribution over next states, conditioned on action a being taken in state s,
• $\Omega(\cdot|s,a) \in \mathcal{P}(\mathcal{O})$ is the probability distribution over possible observations, conditioned on action a being taken to reach state s where the observation is perceived,
• $P_0 \in \mathcal{P}(\mathcal{S})$ is the probability distribution according to which the initial state is selected,
• $R(s,a) \sim q(\cdot|s,a) \in \mathcal{P}(\mathbb{R})$ is a random variable representing the reward obtained when action a is taken in state s.

All assumptions are similar to MDPs, with the addition that $\mathcal{P}(\mathcal{O})$ is the set of probability distributions on (Borel) subsets of $\mathcal{O}$, and Ω is stationary. As a motivation for the POMDP model, we revisit again the online shop domain.

Example 2.2 (Online shop – hidden state setting)**.** Recall the online shop domain of Example 1.1. In a realistic scenario, some features of the customer, such as its gender or age, might not be visible to the decision maker due to privacy, or other reasons. In such a case, the MDP model, which requires the full state information for making decisions is not suitable. In the POMDP model, only observable quantities, such as the items in the shopping cart, are used for making decisions.

Since the state is not directly observed, the agent must rely on the recent history of actions and observations, $\{o_{t+1}, a_t, o_t, \ldots, o_1, a_0\}$

to infer a distribution over states. This *belief* (also called *information state*) is defined over the state probability simplex, $b_t \in \Delta$, and can be calculated recursively as:

$$b_{t+1}(s') = \frac{\Omega(o_{t+1}|s', a_t) \int_{\mathcal{S}} P(s'|s, a_t) b_t(s) ds}{\int_{\mathcal{S}} \Omega(o_{t+1}|s'', a_t) \int_{\mathcal{S}} P(s''|s, a_t) b_t(s) ds ds''} , \tag{2.7}$$

where the sequence is initialized at $b_0 := P_0$ and the denominator can be seen as a simple normalization function. For convenience, we sometimes denote the belief update (Eq. 2.7) as $b_{t+1} = \tau(b_t, a, o)$.

In the POMDP framework, the action-selection policy is defined as $\mu : \Delta(s) \to \mathcal{A}$. Thus, solving a POMDP involves finding the optimal policy, μ^*, that maximizes the expected discounted sum of rewards for all belief states. This can be defined using a variant of the Bellman equation

$$V^*(b_t) = \max_{a \in \mathcal{A}} \left[\int_{\mathcal{S}} R(s, a) b_t(s) ds + \gamma \int_{\mathcal{O}} \Pr(o|b_t, a) V^*(\tau(b_t, a, o)) do \right] . \tag{2.8}$$

The optimal value function for a finite-horizon POMDP can be shown to be piecewise-linear and convex [Smallwood and Sondik, 1973, Porta et al., 2006]. In that case, the value function V_t at any finite planning horizon t can be represented by a finite set of linear segments $\Gamma_t = \{\alpha_0, \alpha_1, \ldots, \alpha_m\}$, often called α-vectors (when $\mathcal{S}$ is discrete) or α-functions (when $\mathcal{S}$ is continuous). Each defines a linear function over the belief state space associated with some action $a \in \mathcal{A}$. The value of a given α_i at a belief b_t can be evaluated by linear interpolation

$$\alpha_i(b_t) = \int_{\mathcal{S}} \alpha_i(s) b_t(s) ds. \tag{2.9}$$

The value of a belief state is the maximum value returned by one of the α-functions for this belief state

$$V_t^*(b_t) = \max_{\alpha \in \Gamma_t} \int_{\mathcal{S}} \alpha(s) b_t(s) . \tag{2.10}$$

In that case, the best action, $\mu^*(b_t)$, is the one associated with the α-vector that returns the best value.

The belief as defined here can be interpreted as a state in a particular kind of MDP, often called the *belief-MDP*. The main intuition is that

for any partially observable system with known parameters, the belief is actually fully observable and computable (for small enough state spaces and is approximable for others). Thus, the planning problem is in fact one of planning in an MDP, where the state space corresponds to the belief simplex. This does not lead to any computational advantage, but conceptually, suggests that known results and methods for MDPs can also be applied to the belief-MDP.

2.4 Reinforcement Learning

Reinforcement learning (RL) [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998] is a class of learning *problems* in which an agent (or controller) interacts with a dynamic, stochastic, and incompletely known environment, with the goal of finding an action-selection strategy, or *policy*, to optimize some measure of its long-term performance. The interaction is conventionally modeled as an MDP, or if the environment state is not always completely observable, as a POMDP [Puterman, 1994].

Now that we have defined the MDP model, the next step is to solve it, i.e., to find an optimal policy.[4] In some cases, MDPs can be solved analytically, and in many cases they can be solved iteratively by *dynamic* or *linear programming* (e.g., see [Bertsekas and Tsitsiklis, 1996]). However, in other cases these methods cannot be applied because either the state space is too large, a system model is available only as a simulator, or no system model is available at all. It is in these cases that RL techniques and algorithms may be helpful.

Reinforcement learning solutions can be viewed as a broad class of sample-based methods for solving MDPs. In place of a model, these methods use sample trajectories of the system and the agent interacting, such as could be obtained from a simulation. It is not unusual in practical applications for such a simulator to be available when an explicit transition-probability model of the sort suitable for use by dynamic or linear programming is not [Tesauro, 1994, Crites and Barto,

[4]It is not possible to find an optimal policy in many practical problems. In such cases, the goal would be to find a reasonably good policy, i.e., a policy with large enough value function.

1998]. Reinforcement learning methods can also be used with no model at all, by obtaining sample trajectories from direct interaction with the system [Baxter et al., 1998, Kohl and Stone, 2004, Ng et al., 2004].

Reinforcement learning solutions can be categorized in different ways. Below we describe two common categorizations that are relevant to the structure of this paper.

Model-based vs. Model-free Methods: Reinforcement learning algorithms that explicitly learn a system model and use it to solve an MDP are called *model-based* methods. Some examples of these methods are the Dyna architecture [Sutton, 1991] and prioritized sweeping [Moore and Atkeson, 1993]. *Model-free* methods are those that do not explicitly learn a system model and only use sample trajectories obtained by direct interaction with the system. These methods include popular algorithms such as Q-learning [Watkins, 1989], SARSA [Rummery and Niranjan, 1994], and LSPI [Lagoudakis and Parr, 2003].

Value Function vs. Policy Search Methods: An important class of RL algorithms are those that first find the optimal value function, and then extract an optimal policy from it. This class of RL algorithms contains *value function* methods, of which some well-studied examples include value iteration (e.g., [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998]), policy iteration (e.g., [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998]), Q-learning [Watkins, 1989], SARSA [Rummery and Niranjan, 1994], LSPI [Lagoudakis and Parr, 2003], and fitted Q-iteration [Ernst et al., 2005]. An alternative approach for solving an MDP is to directly search in the space of policies. These RL algorithms are called *policy search* methods. Since the number of policies is exponential in the size of the state space, one has to resort to meta-heuristic search [Mannor et al., 2003] or to local greedy methods. An important class of policy search methods is that of *policy gradient* algorithms. In these algorithms, the policy is taken to be an arbitrary differentiable function of a parameter vector, and the search in the policy space is directed by the gradient of a performance function

with respect to the policy parameters (e.g., [Williams, 1992, Marbach, 1998, Baxter and Bartlett, 2001]). There is a third class of RL methods that use policy gradient to search in the policy space, and at the same time estimate a value function. These algorithms are called *actor-critic* (e.g., [Konda and Tsitsiklis, 2000, Sutton et al., 2000, Bhatnagar et al., 2007, Peters and Schaal, 2008, Bhatnagar et al., 2009]). They can be thought of as RL analogs of dynamic programming's policy iteration method. Actor-critic methods are based on the simultaneous online estimation of the parameters of two structures, called the *actor* and the *critic*. The actor and the critic correspond to conventional action-selection policy and value function, respectively. These problems are separable, but are solved simultaneously to find an optimal policy.

2.5 Bayesian Learning

In Bayesian learning, we make inference about a random variable X by producing a probability distribution for X. Inferences, such as point and interval estimates, may then be extracted from this distribution. Let us assume that the random variable X is hidden and we can only observe a related random variable Y. Our goal is to infer X from the samples of Y. A simple example is when X is a physical quantity and Y is its noisy measurement. Bayesian inference is usually carried out in the following way:

1. We choose a probability density $P(X)$, called the *prior distribution*, that expresses our beliefs about the random variable X before we observe any data.

2. We select a statistical model $P(Y|X)$ that reflects our belief about Y given X. This model represents the statistical dependence between X and Y.

3. We observe data $Y = y$.

4. We update our belief about X by calculating its posterior distribution using Bayes rule

$$P(X|Y = y) = \frac{P(y|X)P(X)}{\int P(y|X')P(X')dX'} .$$

Assume now that $P(X)$ is parameterized by an unknown vector of parameters $\boldsymbol{\theta}$ in some parameter space Θ; we denote this as $P_{\boldsymbol{\theta}}(X)$. Let $X_1, \ldots, X_n$ be a random i.i.d. sample drawn from $P_{\boldsymbol{\theta}}(X)$. In general, updating the posterior $P_{\boldsymbol{\theta}}(X|Y = y)$ is difficult, due to the need to compute the normalizing constant $\int_\Theta P(y|X')P_{\boldsymbol{\theta}}(X')d\boldsymbol{\theta}$. However, for the case of *conjugate* family distributions, we can update the posterior in closed-form by simply updating the parameters of the distribution. In the next two sections, we consider three classes of conjugate distributions: Beta and Dirichlet distributions and Gaussian Processes (GPs).

2.5.1 Beta and Dirichlet Distributions

A simple example of a conjugate family is the Beta distribution, which is conjugate to the Binomial distribution. Let $Beta(\alpha, \beta)$ be defined by the density function $f(p|\alpha, \beta) \propto p^{\alpha-1}(1-p)^{\beta-1}$ for $p \in [0, 1]$, and parameters $\alpha, \beta \geq 0$. Let $Binomial(n, p)$ be defined by the density function $f(k|n, p) \propto p^k(1-p)^{n-k}$ for $k \in \{0, 1, \ldots, n\}$, and parameters $p \in [0, 1]$ and $n \in \mathbb{N}$. Consider $X \sim Binomial(n, p)$ with unknown probability parameter p, and consider a prior $Beta(\alpha, \beta)$ over the unknown value of p. Then following an observation $X = x$, the posterior over p is also Beta distributed and is defined by $Beta(\alpha + x, \beta + n - x)$.

Now let us consider the multivariate extension of this conjugate family. In this case, we have the Multinomial distribution, whose conjugate is the Dirichlet distribution. Let $X \sim Multinomial_k(p, N)$ be a random variable with unknown probability distribution $p = (p_1, \ldots, p_k)$. The Dirichlet distribution is parameterized by a count vector $\phi = (\phi_1, \ldots, \phi_k)$, where $\phi_i \geq 0$, such that the density of probability distribution $p = (p_1, \ldots, p_k)$ is defined as $f(p|\phi) \propto \prod_{i=1}^k p_i^{\phi_i - 1}$. The distribution $Dirichlet(\phi_1, \ldots, \phi_k)$ can also be interpreted as a prior over the unknown Multinomial distribution p, such that after observing $X = \mathbf{n}$, the posterior over p is also Dirichlet and is defined by $Dirichlet(\phi_1 + n_1, \ldots, \phi_k + n_k)$.

2.5.2 Gaussian Processes and Gaussian Process Regression

A Gaussian process (GP) is an indexed set of jointly Gaussian random variables, i.e., $F(x),\ x \in \mathcal{X}$ is a Gaussian process if and only

if for every finite set of indices $\{x_1, \ldots, x_T\}$ in the index set $\mathcal{X}$, $(F(x_1), \ldots, F(x_T))$ is a vector-valued Gaussian random variable. The GP F may be thought of as a random vector if $\mathcal{X}$ is finite, as a random series if $\mathcal{X}$ is countably infinite, and as a random function if $\mathcal{X}$ is uncountably infinite. In the last case, each instantiation of F is a function $f : \mathcal{X} \to \mathbb{R}$. For a given x, $F(x)$ is a random variable, normally distributed jointly with the other components of F. A GP F can be fully specified by its mean $\bar{f} : \mathcal{X} \to \mathbb{R}$, $\bar{f}(x) = \mathbf{E}[F(x)]$ and covariance $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $k(x, x') = \mathbf{Cov}[F(x), F(x')]$, i.e., $F(\cdot) \sim \mathcal{N}(\bar{f}(\cdot), k(\cdot, \cdot))$. The kernel function $k(\cdot, \cdot)$ encodes our prior knowledge concerning the correlations between the components of F at different points. It may be thought of as inducing a measure of proximity between the members of $\mathcal{X}$. It can also be shown that k defines the function space within which the search for a solution takes place (see [Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004] for more details).

Now let us consider the following generative model:

$$Y(x) = \boldsymbol{H}F(x) + N(x), \tag{2.11}$$

where $\boldsymbol{H}$ is a general linear transformation, F is an unknown function, x is the input, N is a Gaussian noise and independent of F, and Y is the observable process, modeled as a noisy version of $\boldsymbol{H}F$. The objective here is to infer F from samples of Y. Bayesian learning can be applied to this problem in the following way.

1. We choose a prior distribution for F in the form of a GP, $F(\cdot) \sim \mathcal{N}(\bar{f}(\cdot), k(\cdot, \cdot))$. When F and N are Gaussian and independent of each other, the generative model of Eq. 2.11 is known as the *linear statistical model* [Scharf, 1991].

2. The statistical dependence between F and Y is defined by the model-equation (2.11).

3. We observe a sample in the form of $\mathcal{D}_T = \{(x_t, y_t)\}_{t=1}^{T}$.

4. We calculate the posterior distribution of F conditioned on the sample $\mathcal{D}_T$ using the Bayes rule. Below is the process to calculate this posterior distribution.

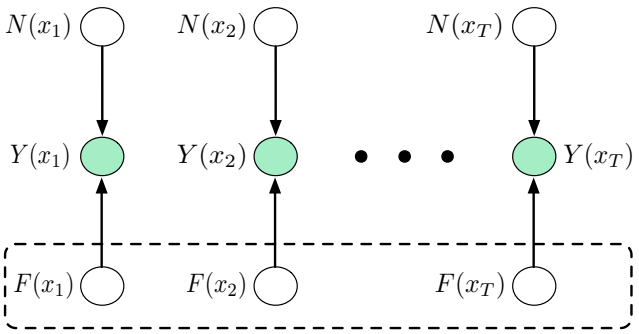

Figure 2.1: A directed graph illustrating the conditional independencies between the latent $F(x_t)$ variables (bottom row), the noise variables $N(x_t)$ (top row), and the observable $Y(x_t)$ variables (middle row), in GP regression (when $\boldsymbol{H} = \boldsymbol{I}$). All of the $F(x_t)$ variables should be interconnected by arrows (forming a clique), due to the dependencies introduced by the prior. To avoid cluttering the diagram, this was marked by the dashed frame surrounding them.

Figure 2.1 illustrates the GP regression setting (when $\boldsymbol{H} = \boldsymbol{I}$) as a graphical model in which arrows mark the conditional dependency relations between the nodes corresponding to the latent $F(x_t)$ and the observed $Y(x_t)$ variables. The model-equation (2.11) evaluated at the training samples may be written as

$$\boldsymbol{Y}_T = \boldsymbol{H}\boldsymbol{F}_T + \boldsymbol{N}_T, \tag{2.12}$$

where $\boldsymbol{F}_T = \big(F(x_1), \ldots, F(x_T)\big)^\top$, $\boldsymbol{Y}_T = (y_1, \ldots, y_T)^\top$, and $\boldsymbol{N}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$. Here $[\boldsymbol{\Sigma}]_{i,j}$ is the measurement noise covariance between the ith and jth samples. In the linear statistical model, we then have $\boldsymbol{F}_T \sim \mathcal{N}(\bar{\boldsymbol{f}}, \boldsymbol{K})$, where $\bar{\boldsymbol{f}} = \big(\bar{f}(x_1), \ldots, \bar{f}(x_T)\big)^\top$ and $[\boldsymbol{K}]_{i,j} = k(x_i, x_j)$ is a $T \times T$ kernel matrix. Since both $\boldsymbol{F}_T$ and $\boldsymbol{N}_T$ are Gaussian and independent from each other, we have $\boldsymbol{Y}_T \sim \mathcal{N}(\boldsymbol{H}\bar{\boldsymbol{f}}, \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma})$. Consider a query point x, we then have

$$\begin{pmatrix} F(x) \\ \boldsymbol{F}_T \\ \boldsymbol{N}_T \end{pmatrix} = \mathcal{N}\left\{ \begin{pmatrix} \bar{f}(x) \\ \bar{\boldsymbol{f}} \\ \boldsymbol{0} \end{pmatrix}, \begin{bmatrix} k(x,x) & \boldsymbol{k}(x)^\top & \boldsymbol{0} \\ \boldsymbol{k}(x) & \boldsymbol{K} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\Sigma} \end{bmatrix} \right\},$$

where $\boldsymbol{k}(x) = \big(k(x_1, x), \ldots, k(x_T, x)\big)^\top$. Using Eq. 2.12, we have the

following transformation:

$$\begin{pmatrix} F(x) \\ \boldsymbol{F}_T \\ \boldsymbol{Y}_T \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \boldsymbol{I} & \boldsymbol{0} \\ 0 & \boldsymbol{H} & \boldsymbol{I} \end{bmatrix} \begin{pmatrix} F(x) \\ \boldsymbol{F}_T \\ \boldsymbol{N}_T \end{pmatrix}, \tag{2.13}$$

where $\boldsymbol{I}$ is the identity matrix. From Eq. 2.13, we have

$$\begin{pmatrix} F(x) \\ \boldsymbol{F}_T \\ \boldsymbol{Y}_T \end{pmatrix} = \mathcal{N}\left\{ \begin{pmatrix} \bar{f}(x) \\ \bar{\boldsymbol{f}} \\ \boldsymbol{H}\bar{\boldsymbol{f}} \end{pmatrix}, \begin{bmatrix} k(x,x) & \boldsymbol{k}(x)^\top & \boldsymbol{k}(x)^\top \boldsymbol{H}^\top \\ \boldsymbol{k}(x) & \boldsymbol{K} & \boldsymbol{K}\boldsymbol{H}^\top \\ \boldsymbol{H}\boldsymbol{k}(x) & \boldsymbol{H}\boldsymbol{K} & \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma} \end{bmatrix} \right\}.$$

Using the Gauss-Markov theorem [Scharf, 1991], we know that $F(x)|\boldsymbol{Y}_T$ (or equivalently $F(x)|\mathcal{D}_T$) is Gaussian, and obtain the following expressions for the posterior mean and covariance of $F(x)$ conditioned on the sample $\mathcal{D}_T$:

$$\begin{aligned} \mathbf{E}[F(x)|\mathcal{D}_T] &= \bar{f}(x) + \boldsymbol{k}(x)^\top \boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1}(\boldsymbol{y}_T - \boldsymbol{H}\bar{\boldsymbol{f}}), \\ \mathbf{Cov}[F(x), F(x')|\mathcal{D}_T] &= k(x,x') - \boldsymbol{k}(x)^\top \boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1}\boldsymbol{H}\boldsymbol{k}(x'), \end{aligned} \tag{2.14}$$

where $\boldsymbol{y}_T = (y_1, \ldots, y_T)^\top$ is one realization of the random vector $\boldsymbol{Y}_T$. It is possible to decompose the expressions in Eq. 2.14 into input dependent terms (which depend on x and x') and terms that only depend on the training samples, as follows:

$$\begin{aligned} \mathbf{E}[F(x)|\mathcal{D}_T] &= \bar{f}(x) + \boldsymbol{k}(x)^\top \boldsymbol{\alpha}, \\ \mathbf{Cov}[F(x), F(x')|\mathcal{D}_T] &= k(x,x') - \boldsymbol{k}(x)^\top \boldsymbol{C}\boldsymbol{k}(x'), \end{aligned} \tag{2.15}$$

where

$$\begin{aligned} \boldsymbol{\alpha} &= \boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1}(\boldsymbol{y}_T - \boldsymbol{H}\bar{\boldsymbol{f}}), \\ \boldsymbol{C} &= \boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1}\boldsymbol{H}. \end{aligned} \tag{2.16}$$

From Eqs. 2.15 and 2.16, we can conclude that $\boldsymbol{\alpha}$ and $\boldsymbol{C}$ are *sufficient statistics* for the posterior moments.

If we set the transformation $\boldsymbol{H}$ to be the identity and assume that the noise terms corrupting each sample are i.i.d. Gaussian, i.e., $\boldsymbol{N}_T \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$, where σ^2 is the variance of each noise term, the linear statistical model is reduced to the *standard linear regression model*. In this

case, the posterior moments of $F(x)$ can be written as

$$\begin{aligned} \mathbf{E}\big[F(x)|\mathcal{D}_T\big] &= \bar{f}(x) + \boldsymbol{k}(x)^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y}_T - \bar{\boldsymbol{f}}) \\ &= \bar{f}(x) + \boldsymbol{k}(x)^\top \boldsymbol{\alpha}, \\ \mathbf{Cov}\big[F(x), F(x')|\mathcal{D}_T\big] &= k(x,x') - \boldsymbol{k}(x)^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}(x') \\ &= k(x,x') - \boldsymbol{k}(x)^\top \boldsymbol{C} \boldsymbol{k}(x'), \end{aligned} \tag{2.17}$$

with

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y}_T - \bar{\boldsymbol{f}}), \qquad \boldsymbol{C} = (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}. \tag{2.18}$$

The GP regression described above is kernel-based and non-parametric. It is also possible to employ a parametric representation under very similar assumptions. In the parametric setting, the GP F is assumed to consist of a linear combination of a finite number n of basis functions $\varphi_i : \mathcal{X} \to \mathbb{R},\ i = 1, \ldots, n$. In this case, F can be written as $F(\cdot) = \sum_{i=1}^n \varphi_i(\cdot) W_i = \boldsymbol{\phi}(\cdot)^\top \boldsymbol{W}$, where $\boldsymbol{\phi}(\cdot) = \big(\varphi_1(\cdot), \ldots, \varphi_n(\cdot)\big)^\top$ is the feature vector and $\boldsymbol{W} = (W_1, \ldots, W_n)^\top$ is the weight vector. The model-equation (2.11) now becomes

$$Y(x) = \boldsymbol{H} \boldsymbol{\phi}(x)^\top \boldsymbol{W} + N(x).$$

In the parametric GP regression, the randomness in F is due to $\boldsymbol{W}$ being a random vector. Here we consider a Gaussian prior over $\boldsymbol{W}$, distributed as $\boldsymbol{W} \sim \mathcal{N}(\bar{\mathbf{w}}, \boldsymbol{S}_{\mathbf{w}})$. By applying the Bayes rule, the posterior (Gaussian) distribution of $\boldsymbol{W}$ conditioned on the observed data $\mathcal{D}_T$ can be computed as

$$\begin{aligned} \mathbf{E}\big[\boldsymbol{W}|\mathcal{D}_T\big] &= \bar{\mathbf{w}} + \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\Phi} \boldsymbol{H}^\top (\boldsymbol{H} \boldsymbol{\Phi}^\top \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\Phi} \boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1} (\boldsymbol{y}_T - \boldsymbol{H} \boldsymbol{\Phi}^\top \bar{\mathbf{w}}), \\ \mathbf{Cov}\big[\boldsymbol{W}|\mathcal{D}_T\big] &= \boldsymbol{S}_{\mathbf{w}} - \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\Phi} \boldsymbol{H}^\top (\boldsymbol{H} \boldsymbol{\Phi}^\top \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\Phi} \boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1} \boldsymbol{H} \boldsymbol{\Phi}^\top \boldsymbol{S}_{\mathbf{w}}, \end{aligned} \tag{2.19}$$

where $\boldsymbol{\Phi} = \big[\boldsymbol{\phi}(x_1), \ldots, \boldsymbol{\phi}(x_T)\big]$ is a $n \times T$ feature matrix. Finally, since $F(x) = \boldsymbol{\phi}(x)^\top \boldsymbol{W}$, the posterior mean and covariance of F can be easily computed as

$$\begin{aligned}
\mathbf{E}\big[F(x)|\mathcal{D}_T\big] &= \boldsymbol{\phi}(x)^\top \bar{\mathbf{w}} \\
&\quad + \boldsymbol{\phi}(x)^\top \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\Phi} \boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{\Phi}^\top \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\Phi} \boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1} (\boldsymbol{y}_T - \boldsymbol{H}\boldsymbol{\Phi}^\top \bar{\mathbf{w}}), \\
\mathbf{Cov}\big[F(x), F(x')|\mathcal{D}_T\big] &= \boldsymbol{\phi}(x)^\top \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\phi}(x') \\
&\quad - \boldsymbol{\phi}(x)^\top \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\Phi} \boldsymbol{H}^\top (\boldsymbol{H}\boldsymbol{\Phi}^\top \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\Phi} \boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1} \boldsymbol{H}\boldsymbol{\Phi}^\top \boldsymbol{S}_{\mathbf{w}} \boldsymbol{\phi}(x').
\end{aligned} \tag{2.20}$$

Similar to the non-parametric setting, for the standard linear regression model with the prior $\boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, Eq. 2.19 may be written as

$$\begin{aligned}
\mathbf{E}\big[\boldsymbol{W}|\mathcal{D}_T\big] &= \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}_T, \\
\mathbf{Cov}\big[\boldsymbol{W}|\mathcal{D}_T\big] &= \boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{\Phi}^\top.
\end{aligned} \tag{2.21}$$

To have a smaller matrix inversion when $T > n$, Eq. 2.21 may be written as

$$\begin{aligned}
\mathbf{E}\big[\boldsymbol{W}|\mathcal{D}_T\big] &= (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{\Phi} \boldsymbol{y}_T, \\
\mathbf{Cov}\big[\boldsymbol{W}|\mathcal{D}_T\big] &= \sigma^2 (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2 \boldsymbol{I})^{-1}.
\end{aligned} \tag{2.22}$$

For more details on GPs and GP regression, see [Rasmussen and Williams, 2006].

3

Bayesian Bandits

This section focuses on Bayesian learning methods for regret minimization in the multi-armed bandits (MAB) model. We review classic performance bounds for this problem and state-of-the-art results for several Bayesian approaches.

In the MAB model (§2.1), the only unknown quantities are the outcome probabilities $P(\cdot|a)$. The idea behind Bayesian approaches to MAB is to use Bayesian inference (§2.5) for learning $P(\cdot|a)$ from the outcomes observed during sequential interaction with the MAB. Following the framework of §2.5, the outcome probabilities are parameterized by an unknown vector $\boldsymbol{\theta}$, and are henceforth denoted by $P_{\boldsymbol{\theta}}(\cdot|a)$. The parameter vector $\boldsymbol{\theta}$ is assumed to be drawn from a prior distribution P_{prior}. As a concrete example, consider a MAB with Bernoulli arms:

Example 3.1 (Bernoulli K-MAB with a Beta prior)**.** Consider a K-MAB where the set of outcomes is binary $\mathcal{Y} = \{0, 1\}$, and the reward is the identity $r(Y) = Y$. Let the outcome probabilities be parameterized by $\boldsymbol{\theta} \in \mathbb{R}^K$, such that $Y(a) \sim \text{Bernoulli}[\boldsymbol{\theta}_a]$. The prior for each $\boldsymbol{\theta}_a$ is $Beta(\alpha_a, \beta_a)$. Following an observation outcome $Y(a) = y$, the posterior for $\boldsymbol{\theta}_a$ is updated to $Beta(\alpha_a + y, \beta_a + y - 1)$. Also note that the (posterior) expected reward for arm a is now $\mathbf{E}\left[r(Y(a))\right] = \mathbf{E}\left[\mathbf{E}\left[Y(a)|\,\boldsymbol{\theta}_a\right]\right] = \mathbf{E}\left[\boldsymbol{\theta}_a\right] = \frac{\alpha_a + y}{\alpha_a + y + \beta_a + y - 1}$.

The principal challenge of Bayesian MAB algorithms, however, is to utilize the posterior $\boldsymbol{\theta}$ for selecting an adequate policy that achieves low regret.

Conceptually, there are several reasons to adopt a Bayesian approach for MABs. First, from a modelling perspective, the prior for $\boldsymbol{\theta}$ is a natural means for embedding prior knowledge, or structure of the problem. Second, the posterior for $\boldsymbol{\theta}$ encodes the uncertainty about the outcome probabilities at each step of the algorithm, and may be used for guiding the exploration to more relevant areas.

A Bayesian point of view may also be taken towards the performance measure of the algorithm. Recall that the performance of a MAB algorithm is measured by its expected regret. This expectation, however, may be defined in two different ways, depending on how the parameters $\boldsymbol{\theta}$ are treated. In the frequentist approach, henceforth termed *'frequentist regret'* and denoted $\mathbf{E}_{\boldsymbol{\theta}}\big[\mathbf{Regret}(T)\big]$, the parameter vector $\boldsymbol{\theta}$ is fixed, and treated as a constant in the expectation. The expectation is then computed with respect to the stochastic outcomes and the action selection policy. On the other hand, in *'Bayesian regret'*, a.k.a. *Bayes risk*, $\boldsymbol{\theta}$ is assumed to be drawn from the prior, and the expectation $\mathbf{E}\big[\mathbf{Regret}(T)\big]$ is over the stochastic outcomes, the action selection rule, and the prior distribution of $\boldsymbol{\theta}$. Note that the optimal action a^* depends on $\boldsymbol{\theta}$. Therefore, in the expectation it is also considered a random variable.

We emphasize the separation of Bayesian MAB algorithms and Bayesian regret analysis. In particular, the performance of a Bayesian MAB algorithm (i.e., an algorithm that uses Bayesian learning techniques) may be measured with respect to a frequentist regret, or a Bayesian one.

3.1 Classical Results

In their seminal work, Lai and Robbins [Lai and Robbins, 1985] proved asymptotically tight bounds on the frequentist regret in terms of the Kullback-Leibler (KL) divergence between the distributions of the rewards of the different arms. These bounds grow logarithmically with

the number of steps T, such that regret is $\mathcal{O}(\ln T)$. Mannor and Tsitsiklis [Mannor and Tsitsiklis, 2004] later showed non-asymptotic lower bounds with a similar logarithmic dependence on T. For the Bayesian regret, the lower bound on the regret is $\mathcal{O}\left(\sqrt{KT}\right)$ (see, e.g., Theorem 3.5 of Bubeck and Cesa-Bianchi [Bubeck and Cesa-Bianchi, 2012]).

In the Bayesian setting, and for models that admit sufficient statistics, Gittins [Gittins, 1979] showed that an optimal strategy may be found by solving a specific MDP planning problem. The key observation here is that the dynamics of the posterior for each arm may be represented by a special MDP termed a *bandit process* [Gittins, 1979].

Definition 3.1 (Bandit Process)**.** A bandit process is an MDP with two actions $\mathcal{A} = \{0, 1\}$. The control 0 freezes the process in the sense that $P(s' = s|s, 0) = 1$, and $R(s, 0) = 0$. Control 1 continues the process, and induces a standard MDP transition to a new state with probability $P(\cdot|s, 1)$, and reward $R(s, 1)$.

For example, consider the case of Bernoulli bandits with a Beta prior as described in Example 3.1. We identify the state s_a of the bandit process for arm a as the posterior parameters $s_a = (\alpha_a, \beta_a)$. Whenever arm a is pulled, the continuation control is applied as follows: we draw some $\boldsymbol{\theta}_a$ from the posterior, and then draw an outcome $Y \sim \text{Bernoulli}[\boldsymbol{\theta}_a]$. The state subsequently transitions to an updated posterior (cf. Example 3.1) $s'_a = (\alpha_a + Y, \beta_a + Y - 1)$, and a reward of $\mathbf{E}[r(Y)]$ is obtained, where the expectation is taken over the posterior.

The K-MAB problem, thus, may be seen as a particular instance of a general model termed *simple family of alternative bandit processes* (SFABP) [Gittins, 1979].

Model 5 (Simple Family of Alternative Bandit Processes) A simple family of alternative bandit processes is a set of K bandit processes. For each bandit process $i \in 1, \ldots, K$ we denote by s_i, $P_i(\cdot|s_i, 1)$, and $R_i(s_i, 1)$ the corresponding state, transition probabilities, and reward, respectively. At each time $t = 1, 2, \ldots$, a single bandit $i_t \in 1, \ldots, K$ that is in state $s_{i_t}(t)$ is activated, by applying to it the continuation control, and all other bandits are frozen. The objective is to find a bandit selection policy that maximizes the expected total

γ-discounted reward

$$\mathbf{E}\left[\sum_{t=0}^{\infty}\gamma^t R_{i_t}(s_{i_t}(t),1)\right].$$

An important observation is that for the K-MAB problem, as defined above, the SFABP performance measure captures an expectation of the total discounted reward with respect to the *prior* distribution of the parameters $\boldsymbol{\theta}$. In this sense, this approach is a *full* Bayesian K-MAB solution (cf. the Bayesian regret vs. frequentist regret discussion above). In particular, note that the SFABP performance measure implicitly balances exploration and exploitation. To see this, recall the Bernoulli bandits example, and note that the reward of each bandit process is the *posterior* expected reward of its corresponding arm. Since the posterior distribution is encoded in the process state, future rewards in the SFABP performance measure depend on the future state of knowledge, thereby implicitly quantifying the value of additional observations for each arm.

The SFABP may be seen as a single MDP, with a state $(s_1, s_2, \ldots, s_K)$ that is the conjunction of the K individual bandit process states. Thus, naively, an optimal policy may be computed by solving this MDP. Unfortunately, since the size of the state space of this MDP is exponential in K, such a naive approach is intractable. The virtue of the celebrated Gittins index theorem [Gittins, 1979], is to show that this problem nevertheless has a tractable solution.

Theorem 3.1. [Gittins, 1979] The objective of SFABP is maximized by following a policy that always chooses the bandit with the largest Gittins index

$$G_i(s_i) = \sup_{\tau\geq 1}\frac{\mathbf{E}\left[\sum_{t=0}^{\tau-1}\gamma^t R_i(s_i(t),1)\middle|\, s_i(0)=s_i\right]}{\mathbf{E}\left[\sum_{t=0}^{\tau-1}\gamma^t\middle|\, s_i(0)=s_i\right]},$$

where τ is any (past measurable) stopping time.

The crucial advantage of Theorem 3.1, is that calculating G_i may be done *separately* for each arm, thereby avoiding the exponential complexity in K. The explicit calculation, however, is technically involved,

and beyond the scope of this survey. For reference see [Gittins, 1979], and also [Tsitsiklis, 1994] for a simpler derivation, and [Niño-Mora, 2011] for a finite horizon setting.

Due to the technical complexity of calculating optimal Gittin's index policies, recent approaches concentrate on much simpler algorithms, that nonetheless admit optimal upper bounds (i.e., match the order of their respective lower bounds up to constant factors) on the expected regret.

3.2 Bayes-UCB

The upper confidence bound (UCB) algorithm [Auer et al., 2002] is a popular frequentist approach for MABs that employs an 'optimistic' policy to reduce the chance of overlooking the best arm. The algorithm starts by playing each arm once. Then, at each time step t, UCB plays the arm a that maximizes $< r_a > + \sqrt{\frac{2 \ln t}{t_a}}$, where $< r_a >$ is the average reward obtained from arm a, and t_a is the number of times arm a has been playing so far. The optimistic upper confidence term $\sqrt{\frac{2 \ln t}{t_a}}$ guarantees that the empirical average does not underestimate the best arm due to 'unlucky' reward realizations.

The Bayes-UCB algorithm of Kaufmann et al. [Kaufmann et al., 2012a] extends the UCB approach to the Bayesian setting. A posterior distribution over the expected reward of each arm is maintained, and at each step, the algorithm chooses the arm with the maximal posterior $(1 - \beta_t)$-quantile, where β_t is of order $1/t$. Intuitively, using an upper quantile instead of the posterior mean serves the role of 'optimism', in the spirit of the original UCB approach. For the case of Bernoulli distributed outcomes and a uniform prior on the reward means, Kaufmann et al. [Kaufmann et al., 2012a] prove a *frequentist* upper bound on the expected regret that matches the lower bound of Lai and Robbins [Lai and Robbins, 1985].

3.3 Thompson Sampling

The Thompson Sampling algorithm (TS) suggests a natural Bayesian approach to the MAB problem using randomized probability match-

ing [Thompson, 1933]. Let P_{post} denote the posterior distribution of $\boldsymbol{\theta}$ given the observations up to time t. In TS, at each time step t, we sample a parameter $\hat{\boldsymbol{\theta}}$ from the posterior and select the optimal action with respect to the model defined by $\hat{\boldsymbol{\theta}}$. Thus, effectively, we match the action selection probability to the posterior probability of each action being optimal. The outcome observation is then used to update the posterior P_{post}.

Algorithm 1 Thompson Sampling

1: **TS**(P_{prior})
   • P_{prior} prior distribution over $\boldsymbol{\theta}$
2: $P_{\text{post}} := P_{\text{prior}}$
3: **for** $t = 1, 2, \ldots$ **do**
4: Sample $\hat{\boldsymbol{\theta}}$ from P_{post}
5: Play arm $a_t = \arg\max_{a \in \mathcal{A}} \mathbf{E}_{y \sim P_{\hat{\boldsymbol{\theta}}}(\cdot|a)} [r(y)]$
6: Observe outcome Y_t and update P_{post}
7: **end for**

Recently, the TS algorithm has drawn considerable attention due to its state-of-the-art empirical performance [Scott, 2010, Chapelle and Li, 2011], which also led to its use in several industrial applications [Graepel et al., 2010, Tang et al., 2013]. We survey several theoretical studies that confirm TS is indeed a sound MAB method with state-of-the-art performance guarantees.

We mention that the TS idea is not limited to MAB problems, but can be seen as a general sampling technique for Bayesian learning, and has been applied also to contextual bandit and RL problems, among others [Strens, 2000, Osband et al., 2013, Abbasi-Yadkori and Szepesvari, 2015]. In this section, we present results for the MAB and contextual bandit settings, while the RL related results are given in Chapter 4.

Agrawal and Goyal [Agrawal and Goyal, 2012] presented *frequentist* regret bounds for TS. Their analysis is specific to Bernoulli arms with a uniform prior, but they show that by a clever modification of the algorithm it may also be applied to general arm distributions. Let $\Delta_a = \mathbf{E}_{\boldsymbol{\theta}}[r(Y(a^*)) - r(Y(a))]$ denote the difference in expected reward

between the optimal arm and arm a, when the parameter is $\boldsymbol{\theta}$.

Theorem 3.2. [Agrawal and Goyal, 2012] For the K-armed stochastic bandit problem, the TS algorithm has (frequentist) expected regret

$$\mathbf{E}_{\boldsymbol{\theta}}[\mathbf{Regret}(T)] \leq \mathcal{O}\left(\left(\sum_{a \neq a^*} \frac{1}{\Delta_a^2}\right)^2 \ln T\right).$$

Improved upper bounds were later presented [Kaufmann et al., 2012b] for the specific case of Bernoulli arms with a uniform prior. These bounds are (order) optimal, in the sense that they match the lower bound of [Lai and Robbins, 1985]. Let $\mathbf{KL}_{\boldsymbol{\theta}}(a, a^*)$ denote the Kullback-Leibler divergence between Bernoulli distributions with parameters $\mathbf{E}_{\boldsymbol{\theta}}[r(Y(a))]$ and $\mathbf{E}_{\boldsymbol{\theta}}[r(Y(a^*))]$.

Theorem 3.3. [Kaufmann et al., 2012b] For any $\epsilon > 0$, there exists a problem-dependent constant $C(\epsilon, \boldsymbol{\theta})$ such that the regret of TS satisfies

$$\mathbf{E}_{\boldsymbol{\theta}}[\mathbf{Regret}(T)] \leq (1+\epsilon) \sum_{a \neq a^*} \frac{\Delta_a(\ln(T) + \ln\ln(T))}{\mathbf{KL}_{\boldsymbol{\theta}}(a, a*)} + C(\epsilon, \boldsymbol{\theta}).$$

More recently, Agrawal and Goyal [Agrawal and Goyal, 2013a] showed a *problem independent* frequentist regret bound of order $\mathcal{O}\left(\sqrt{KT\ln T}\right)$, for the case of Bernoulli arms with a Beta prior. This bound holds *for all* $\boldsymbol{\theta}$, and implies that a Bayesian regret bound with a similar order holds; up to the logarithmic factor, this bound is order-optimal.

Liu and Li [Liu and Li, 2015] investigated the importance of having an *informative* prior in TS. Consider the special case of two-arms and two-models, i.e., $K = 2$, and $\boldsymbol{\theta} \in \{\theta_{\text{true}}, \theta_{\text{false}}\}$, and assume that θ_{true} is the true model parameter. When $P_{\text{prior}}(\theta_{\text{true}})$ is small, the *frequentist* regret of TS is upper-bounded by $\mathcal{O}\left(\sqrt{\frac{T}{P_{\text{prior}}(\theta_{\text{true}})}}\right)$, and when $P_{\text{prior}}(\theta_{\text{true}})$ is sufficiently large, the regret is upper-bounded by $\mathcal{O}\left(\sqrt{(1 - P_{\text{prior}}(\theta_{\text{true}}))T}\right)$ [Liu and Li, 2015]. For the special case of two-arms and two-models, regret lower-bounds of matching orders are also provided in [Liu and Li, 2015]. These bounds show that an informative prior, i.e., a large $P_{\text{prior}}(\theta_{\text{true}})$, significantly impacts the regret.

One appealing property of TS is its natural extension to cases with structure or dependence between the arms. For example, consider the following extension of the online shop domain:

Example 3.2 (Online Shop with Multiple Product Suggestions)**.** Consider the bandit setting of the online shop domain, as in Example 2.1. Instead of presenting to the customer a single product suggestion, the decision-maker may now suggest M different product suggestions $a_1, \ldots, a_M$ from a pool of suggestions $\mathcal{I}$. The customer, in turn, decides whether or not to buy each product, and the reward is the sum of profits from all items bought.

Naively, this problem may be formalized as a MAB with the action set, $\mathcal{A}$, being the set of all possible combinations of M elements from $\mathcal{I}$. In such a formulation, it is clear that the outcomes for actions with overlapping product suggestions are correlated.

In TS, such dependencies between the actions may be incorporated by simply updating the posterior. Recent advances in Bayesian inference such as particle filters and Markov Chain Monte-Carlo (MCMC) provide efficient numerical procedures for complex posterior updates. Gopalan et al. [Gopalan et al., 2014] presented frequentist high-probability regret bounds for TS with general reward distributions and priors, and correlated actions. Let $N_T(a)$ denote the play count of arm a until time T and let $D_{\hat{\boldsymbol{\theta}}} \in \mathbb{R}^{|\mathcal{A}|}$ denote a vector of Kullback-Leibler divergences $D_{\hat{\boldsymbol{\theta}}}(a) = \mathbf{KL}\left(P_{\boldsymbol{\theta}}(\cdot|a) \| P_{\hat{\Theta}}(\cdot|a)\right)$, where $\boldsymbol{\theta}$ is the true model in a frequentist setting. For each action $a \in \mathcal{A}$, we define S_a to be the set of model parameters $\boldsymbol{\theta}$ in which the optimal action is a. Within S_a, let S'_a be the set of models $\hat{\boldsymbol{\theta}}$ that exactly match the true model $\boldsymbol{\theta}$ in the sense of the marginal outcome distribution for action a: $\mathbf{KL}\left(P_{\boldsymbol{\theta}}(\cdot|a) \| P_{\hat{\boldsymbol{\theta}}}(\cdot|a)\right) = 0$. Moreover, let $e^{(j)}$ denote the j-th unit vector in a finite-dimensional Euclidean space. The following result holds under the assumption of finite action and observation spaces, and that the prior has a finite support with a positive probability for the true model $\boldsymbol{\theta}$ ('grain of truth' assumption).

Theorem 3.4. [Gopalan et al., 2014] For $\delta, \epsilon \in (0, 1)$, there exists $T^* > 0$ such that for all $T > T^*$, with probability at least $1 - \delta$,

$$\sum_{a \neq a^*} N_T(a) \leq B + C(\log T),$$

where $B \equiv B(\delta, \epsilon, \mathcal{A}, \mathcal{Y}, \boldsymbol{\theta}, P_{\text{prior}})$ is a problem-dependent constant that

does not depend on T and

$$\begin{aligned} C(\log T) := \max \quad & \sum_{k=1}^{K-1} z_k(a_k) \\ s.t. \quad & z_k \in \mathbb{N}_+^{K-1} \times \{0\}, \quad a_k \in \mathcal{A} \setminus \{a^*\}, \quad k < K, \\ & z_i \succeq z_k, \quad z_i(a_k) = z_k(a_k), \quad i \geq k, \\ & \forall j \geq 1, k \leq K-1 : \\ & \qquad \min_{\hat{\boldsymbol{\theta}} \in S'_{a_k}} \langle z_k, D_{\hat{\boldsymbol{\theta}}} \rangle \geq \frac{1+\epsilon}{1-\epsilon} \log T, \\ & \qquad \min_{\hat{\boldsymbol{\theta}} \in S'_{a_k}} \left\langle z_k - e^{(j)}, D_{\hat{\boldsymbol{\theta}}} \right\rangle \geq \frac{1+\epsilon}{1-\epsilon} \log T. \end{aligned}$$

As Gopalan et al. [Gopalan et al., 2014] explain, the bound in Theorem 3.4 accounts for the dependence between the arms, and thus, provides tighter guarantees when there is information to be gained from this dependence. For example, in the case of selecting subsets of M arms described earlier, calculating the term $C(\log T)$ in Theorem 3.4 gives a bound of $\mathcal{O}\big((K-M)\log T\big)$, even though the total number of actions is $\binom{K}{M}$.

We proceed with an analysis of the Bayesian regret of TS. Using information theoretic tools, Russo and Van Roy [Russo and Van Roy, 2014a] elegantly bound the Bayesian regret, and investigate the benefits of having an informative prior. Let p_t denote the distribution of the selected arm at time t. Note that by the definition of the TS algorithm, p_t encodes the posterior distribution that the arm is optimal. Let $p_{t,a}(\cdot)$ denote the *posterior* outcome distribution at time t, when selecting arm a, and let $p_{t,a}(\cdot|a^*)$ denote the posterior outcome distribution at time t, when selecting arm a, conditioned on the event that a^* is the optimal action. Both $p_{t,a}(\cdot)$ and $p_{t,a}(\cdot|a^*)$ are random and depend on the prior and on the history of actions and observations up to time t. A key quantity in the analysis of [Russo and Van Roy, 2014a] is the *information ratio* defined by

$$\Gamma_t = \frac{\mathbf{E}_{a \sim p_t}\left[\mathbf{E}_{y \sim p_{t,a}(\cdot|a)} r(y) - \mathbf{E}_{y \sim p_{t,a}(\cdot)} r(y)\right]}{\sqrt{\mathbf{E}_{a \sim p_t, a^* \sim p_t}\left[\mathbf{KL}\big(p_{t,a}(\cdot|a^*) \| p_{t,a}(\cdot)\big)\right]}}. \tag{3.1}$$

Note that Γ_t is also random and depends on the prior and on the history of the algorithm up to time t. As Russo and Van Roy [Russo and Van Roy, 2014a] explain, the numerator in Eq. 3.1 roughly captures how much knowing that the selected action is optimal influences the expected reward observed, while the denominator measures how much on average, knowing which action is optimal changes the observations at the selected action. Intuitively, the information ratio tends to be small when knowing which action is optimal significantly influences the anticipated observations at many other actions.

The following theorem relates a bound on Γ_t to a bound on the Bayesian regret.

Theorem 3.5. [Russo and Van Roy, 2014a] For any $T \in \mathbb{N}$, if $\Gamma_t \leq \bar{\Gamma}$ almost surely for each $t \in \{1, \ldots, T\}$, the TS algorithm satisfies

$$\mathbf{E}[\mathbf{Regret}(T)] \leq \bar{\Gamma}\sqrt{\mathbf{H}(p_1)T},$$

where $\mathbf{H}(\cdot)$ denotes the Shannon entropy.

Russo and Van Roy [Russo and Van Roy, 2014a] further showed that in general, $\Gamma_t \leq \sqrt{K/2}$, giving an order-optimal upper bound of $\mathcal{O}\left(\sqrt{\frac{1}{2}\mathbf{H}(p_1)KT}\right)$. However, structure between the arms may be exploited to further bound the information ratio more tightly. For example, consider the case of linear optimization under bandit feedback where we have $\mathcal{A} \subset \mathbb{R}^d$, and the reward satisfies $\mathbf{E}_{y \sim P_{\boldsymbol{\theta}}(\cdot|a)}[r(y)] = a^\top \boldsymbol{\theta}$. In this case, an order-optimal bound of $\mathcal{O}\left(\sqrt{\frac{1}{2}\mathbf{H}(p_1)dT}\right)$ holds [Russo and Van Roy, 2014a]. It is important to note that the term $\mathbf{H}(p_1)$ is bounded by $\log(K)$, but in fact may be much smaller when there is an *informative* prior available.

An analysis of TS with a slightly different flavour was given by Guha and Munagala [Guha and Munagala, 2014], who studied the *stochastic regret* of TS, defined as the expected number of times a sub-optimal arm is chosen, where the expectation is Bayesian, i.e., taken with respect to $P_{\text{prior}}(\boldsymbol{\theta})$. For some horizon T, and prior P_{prior}, let $OPT(T, P_{\text{prior}})$ denote the stochastic regret of an optimal policy. Such a policy exists, and in principle may be calculated using dynamic programming (cf. the Gittins index discussion above). For the cases of two-armed

bandits, and K-MABs with Bernoulli point priors, Guha and Munagala [Guha and Munagala, 2014] show that the stochastic regret of TS, labeled $TS(T, P_{\text{prior}})$ is a 2-approximation of $OPT(T, P_{\text{prior}})$, namely $TS(T, P_{\text{prior}}) \leq 2OPT(T, P_{\text{prior}})$. Interestingly, and in contrast to the asymptotic regret results discussed above, this result holds *for all* T.

We conclude by noting that contextual bandits may be approached using Bayesian techniques in a very similar manner to the MAB algorithms described above. The only difference is that the unknown vector $\boldsymbol{\theta}$ should now parameterize the distribution over actions and context, $P_{\boldsymbol{\theta}}(\cdot|a, s)$. Empirically, the efficiency of TS was demonstrated in an online-advertising application of contextual bandits [Chapelle and Li, 2011]. On the theoretical side, Agrawal and Goyal [Agrawal and Goyal, 2013b] study contextual bandits with rewards that linearly depend on the context, and show a frequentist regret bound of $\tilde{\mathcal{O}}\left(\frac{d^2}{\epsilon}\sqrt{T^{1+\epsilon}}\right)$, where d is the dimension of the context vector, and ϵ is an algorithm-parameter that can be chosen in $(0, 1)$. For the same problem, Russo and Van Roy [Russo and Van Roy, 2014b] derive a Bayesian regret bound of order $\mathcal{O}\left(d\sqrt{T}\ln T\right)$, which, up to logarithmic terms, matches the order of the $\mathcal{O}\left(d\sqrt{T}\right)$ lower bound for this problem [Rusmevichientong and Tsitsiklis, 2010].

4

Model-based Bayesian Reinforcement Learning

This section focuses on Bayesian learning methods that explicitly maintain a posterior over the model parameters and use this posterior to select actions. Actions can be selected both for *exploration* (i.e., achieving a better posterior), or *exploitation* (i.e., achieving maximal return). We review basic representations and algorithms for model-based approaches, both in MDPs and POMDPs. We also present approaches based on sampling of the posterior, some of which provide finite sample guarantees on the learning process.

4.1 Models and Representations

Initial work on model-based Bayesian RL appeared in the control literature, under the topic of Dual Control [Feldbaum, 1961, Filatov and Unbehauen, 2000]. The goal here is to explicitly represent uncertainty over the model parameters, P, q, as defined in §2.2. One way to think about this problem is to see the parameters as unobservable states of the system, and to cast the problem of planning in an MDP with unknown parameters as planning under uncertainty using the POMDP formulation. In this case, the belief tracking operation of Eq. 2.7 will

keep a joint posterior distribution over model parameters and the true physical state of the system, and a policy will be derived to select optimal actions with respect to this posterior.

Let $\theta_{s,a,s'}$ be the (unknown) probability of transitioning from state s to state s' when taking action a, $\theta_{s,a,r}$ the (unknown) probability of obtaining reward r when taking action a at state s, and $\theta \in \Theta$ the set of all such parameters. The belief $P_0(\theta)$ expresses our initial knowledge about the model parameters. We can then compute $b_t(\theta)$, the belief after a t-step trajectory $\{s_{t+1}, r_t, a_t, s_t, \ldots, s_1, r_0, a_0, s_0\}$. Considering a single observation (s, a, r, s'), we have

$$b_{t+1}(\theta') = \eta \Pr(s', r|s, a, \theta') \int_{\mathcal{S},\Theta} \Pr(s', \theta'|s, a, \theta) b_t(\theta) ds d\theta \ , \qquad (4.1)$$

where η is the normalizing factor. It is common to assume that the uncertainty between the parameters is independent, and thus compute $b_t(\theta) = \prod_{s,a} b_t(\theta_{s,a,s'}) b_t(\theta_{s,a,r})$.

Recalling that a POMDP can be represented as a belief-MDP (see §2.3), a convenient way to interpret Eq. 4.1 is as an MDP where the state is defined to be a belief over the unknown parameters. Theoretically, optimal planning in this representation can be achieved using MDP methods for continuous states, the goal being to express a policy over the belief space, and thus, over any posterior over model parameters. From a computational perspective, this is not necessarily a useful objective. First, it is computationally expensive, unless there are only a few unknown model parameters. Second, it is unnecessarily hard: indeed, the goal is to *learn* (via the acquisition of data) an optimal policy that is robust to parameter uncertainty, not necessarily to pre-compute a policy for any possible parameterization. An alternative approach is to make specific assumptions about the structural form of the uncertainty over the model parameters, thereby allowing us to consider an alternate representation of the belief-MDP, one that is mathematically equivalent (under the structural assumptions), but more amenable to computational analysis.

An instance of this type of approach is the Bayes-Adaptive MDP (BAMDP) model [Duff, 2002]. Here we restrict our attention to MDPs with discrete state and action sets. In this case, transition probabilities

consist of Multinomial distributions. The posterior over the transition function can therefore be represented using a Dirichlet distribution, $P(\cdot|s,a) \sim \phi_{s,a}$. For now, we assume that the reward function is known; extensions to unknown rewards are presented in §4.7.

Model 6 (Bayes-Adaptive MDP) Define a Bayes-Adaptive MDP $\mathcal{M}$ to be a tuple $\langle \mathcal{S}', \mathcal{A}, P', P'_0, R' \rangle$ where
• $\mathcal{S}'$ is the set of hyper-states, $\mathcal{S} \times \Phi$,
• $\mathcal{A}$ is the set of actions,
• $P'(\cdot|s,\phi,a)$ is the transition function between hyper-states, conditioned on action a being taken in hyper-state (s,ϕ),
• $P'_0 \in \mathcal{P}(\mathcal{S} \times \Phi)$ combines the initial distribution over physical states, with the prior over transition functions ϕ_0,
• $R'(s,\phi,a) = R(s,a)$ represents the reward obtained when action a is taken in state s.

The BAMDP is defined as an extension of the conventional MDP model. The state space of the BAMDP combines jointly the initial (physical) set of states $\mathcal{S}$, with the posterior parameters on the transition function Φ. We call this joint space the *hyper-state*. Assuming the posterior on the transition is captured by a Dirichlet distribution, we have $\Phi = \{\phi_{s,a} \in \mathbb{N}^{|\mathcal{S}|}, \forall s, a \in \mathcal{S} \times A\}$. The action set is the same as in the original MDP. The reward function is assumed to be known for now, so it depends only on the physical (real) states and actions.

The transition model of the BAMDP captures transitions between hyper-states. Due to the nature of the state space, this transition function has a particular structure. By the chain rule, $\Pr(s', \phi'|s,a,\phi) = \Pr(s'|s,a,\phi)\Pr(\phi'|s,a,s',\phi)$. The first term can be estimated by taking the expectation over all possible transition functions to yield $\frac{\phi_{s,a,s'}}{\sum_{s'' \in \mathcal{S}} \phi_{s,a,s''}}$. For the second term, since the update of the posterior ϕ to ϕ' is deterministic, $\Pr(\phi'|s,a,s',\phi)$ is 1 if $\phi'_{s,a,s'} = \phi_{s,a,s'} + 1$, and 0, otherwise. Because these terms depend only on the previous hyper-state $(s, \phi_{s,a})$ and action a, transitions between hyper-states preserve the Markov property. To summarize, we have (note that $\mathbb{I}(\cdot)$ is the

indicator function):

$$P'(s', \phi' | s, a, \phi) = \frac{\phi_{s,a,s'}}{\sum_{s'' \in \mathcal{S}} \phi_{s,a,s''}} \mathbb{I}(\phi'_{s,a,s'} = \phi_{s,a,s'} + 1). \tag{4.2}$$

It is worth considering the number of states in the BAMDP. Initially (at $t = 0$), there are only $|\mathcal{S}|$, one per real MDP, state (we assume a single prior ϕ_0 is specified). Assuming a fully connected state space in the underlying MDP (i.e., $P(s'|s, a) > 0, \forall s, a$), then at $t = 1$ there are already $|\mathcal{S}| \times |\mathcal{S}|$ states, since $\phi \to \phi'$ can increment the count of any one of its $|\mathcal{S}|$ components. So at horizon t, there are $|\mathcal{S}|^t$ reachable states in the BAMDP. There are clear computational challenges in computing an optimal policy over all such beliefs.

Computational concerns aside, the value function of the BAMDP can be expressed using the Bellman equation

$$\begin{aligned} V^*(s, \phi) &= \max_{a \in \mathcal{A}} \left[R'(s, \phi, a) + \gamma \sum_{(s', \phi') \in \mathcal{S}'} P'(s', \phi' | s, \phi, a) V^*(s', \phi') \right] \\ &= \max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \sum_{s' \in \mathcal{S}} \frac{\phi^a_{s,s'}}{\sum_{s'' \in \mathcal{S}} \phi^a_{s,s''}} V^*(s', \phi') \right]. \end{aligned} \tag{4.3}$$

Let us now consider a simple example, the Chain problem, which is used extensively for empirical demonstrations throughout the literature on model-based BRL.

Example 4.1 (The Chain problem)**.** The 5-state Chain problem [Strens, 2000], shown in Figure 4.1, requires the MDP agent to select between two abstract actions $\{1, 2\}$. Action 1 causes the agent to move to the right with probability 0.8 (effect "a" in Figure 4.1) and causes the agent to reset to the initial state with probability 0.2 (effect "b" in Figure 4.1). Action 2 causes the agent to reset with probability 0.8 (effect "b") and causes the agent to move to the right with probability 0.2 (effect "a"). The action b has constant reward of +2. Rewards vary based on the state and effect ("a" and "b"), as shown in Figure 4.1. The optimal policy is to always choose action 1, causing the agent to potentially receive +10 several times until slipping back (randomly) to

the initial state. Of course if the transition probabilities and rewards are not known, the agent has to trade-off exploration and exploitation to learn this optimal policy.

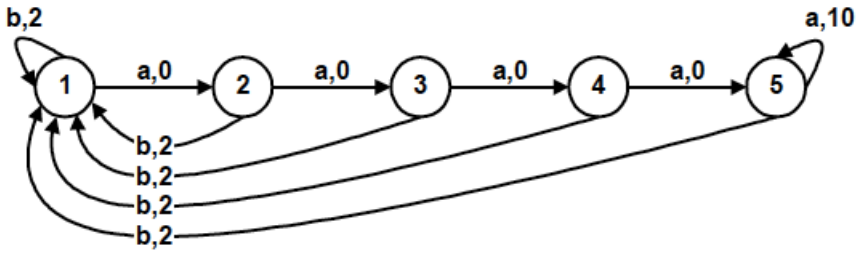

Figure 4.1: The Chain problem (figure reproduced from [Strens, 2000])

4.2 Exploration/Exploitation Dilemma

A key aspect of reinforcement learning is the issue of *exploration*. This corresponds to the question of determining how the agent should choose actions while learning about the task. This is in contrast to the phase called *exploitation*, through which actions are selected so as to maximize expected reward with respect to the current value function estimate.

In the Bayesian RL framework, exploration and exploitation are naturally balanced in a coherent mathematical framework. Policies are expressed over the full information state (or belief), including over model uncertainty. In that framework, the optimal Bayesian policy will be to select actions based on how much reward they yield, but also based on how much information they provide about the parameters of the domain, information which can then be leveraged to acquire even more reward.

Definition 4.1 (Bayes optimality)**.** Let

$$V_t^*(s,\phi) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \int_{\mathcal{S},\Phi} P(s'|b,s,a) V_{t-1}^*(s',\phi') ds' d\theta' \right]$$

be the optimal value function for a t-step planning horizon in a BAMDP.

Any policy that maximizes this expression is called a *Bayes-optimal* policy. In general, a Bayes-optimal policy is mathematically lower than the optimal policy (Eq. 2.6), because it may require additional actions to acquire information about the model parameters. In the next section, we present planning algorithms that seek the Bayes-optimal policy; most are based on heuristics or approximations due to the computation complexity of the problem. In the following section, we also review algorithms that focus on a slightly different criteria, namely efficient (polynomial) learning of the optimal value function.

4.3 Offline Value Approximation

We now review various classes of approximate algorithms for estimating the value function in the BAMDP. We begin with *offline* algorithms that compute the policy *a priori*, for any possible state and posterior. The goal is to compute an action-selection strategy that optimizes the expected return over the hyper-state of the BAMDP. Given the size of the state space in the BAMDP, this is clearly intractable for most domains. The interesting problem then is to devise approximate algorithms that leverage structural constraints to achieve computationally feasible solutions.

Finite-state controllers

Duff [Duff, 2001] suggests using Finite-State Controllers (FSC) to compactly represent the optimal policy μ^* of a BAMDP, and then finding the best FSC in the space of FSCs of some bounded size.

Definition 4.2 (Finite-State Controller)**.** A finite state controller for a BAMDP is defined as a graph, where nodes represent memory states and edges represent observations in the form of (s, a, s') triplets. Each node is also associated with an action (or alternately a distribution over actions), which represents the policy itself.

In this representation, memory states correspond to finite trajectories, rather than full hyper-states. Tractability is achieved by limiting the number of memory states that are included in the graph. Given

a specific finite-state controller (i.e., a compact policy), its expected value can be calculated in closed form using a system of Bellman equations, where the number of equations/variables is equal to $|\mathcal{S}| \times |Q|$, where $|\mathcal{S}|$ is the number of states in the original MDP, and $|Q|$ is the number of memory states in the FSC [Duff, 2001]. The remaining challenge is to optimize the policy. This is achieved using a gradient descent algorithm. A Monte-Carlo gradient estimation is proposed to make it more tractable. This approach presupposes the existence of a good FSC representation for the policy. In general, while conceptually and mathematically straight-forward, this method is computationally feasible only for small domains with few memory states. For many real-world domains, the number of memory states needed to achieve good performance is far too large.

Bayesian Exploration Exploitation Tradeoff in LEarning (BEETLE)

An alternate approximate offline algorithm to solve the BAMDP is called BEETLE [Poupart et al., 2006]. This is an extension of the Perseus algorithm [Spaan and Vlassis, 2005] that was originally designed for conventional POMDP planning, to the BAMDP model. Essentially, at the beginning, hyper-states (s, ϕ) are sampled from random interactions with the BAMDP model. An equivalent continuous POMDP (over the space of states and transition functions) is solved instead of the BAMDP (assuming $b = (s, \phi)$ is a belief state in that POMDP). The value function is represented by a set of *α-functions* over the continuous space of transition functions (see Eqs. 2.8-2.10). In the case of BEETLE, it is possible to maintain a separate set of α-functions for each MDP state, denoted Γ^s. Each α-function is constructed as a linear combination of basis functions $\alpha(b_t) = \int_\theta \alpha(\theta) b(\theta) d\theta$. In practice, the sampled hyper-states can serve to select the set of basis functions θ. The set of α-functions can then be constructed incrementally by applying Bellman updates at the sampled hyper-states using standard point-based POMDP methods [Spaan and Vlassis, 2005, Pineau et al., 2003].

Thus, the constructed α-functions can be shown to be multi-variate polynomials. The main computational challenge is that the number of

terms in the polynomials increases exponentially with the planning horizon. This can be mitigated by projecting each α-function into a polynomial with a smaller number of terms (using basis functions as mentioned above). The method has been tested experimentally in some small simulation domains. The key to applying it in larger domains is to leverage knowledge about the structure of the domain to limit the parameter inference to a few key parameters, or by using parameter tying (whereby a subset of parameters are constrained to have the same posterior).

4.4 Online near-myopic value approximation

We recall from §4.1 that for a planning horizon of t steps, an offline BAMDP planner will consider optimal planning at $|\mathcal{S}|^t$ states. In practice, there may be many fewer states, in particular because some trajectories will not be observed. Online planning approaches interleave planning and execution on a step-by-step basis, so that planning resources are focused on those states that have been observed during actual trajectories. We now review a number of online algorithms developed for the BAMDP framework.

Bayesian dynamic programming

A simple approach, closely related to Thompson sampling (§3.3), was proposed by Sterns [Strens, 2000]. The idea is to sample a model from the posterior distribution over parameters, solve this model using dynamic programming techniques, and use the solved model to select actions. Models are re-sampled periodically (e.g., at the end of an episode or after a fixed number of steps). The approach is simple to implement and does not rely on any heuristics. Goal-directed exploration is achieved via sampling of the models. Convergence to the optimal policy is achievable because the method samples models from the full posterior over parameter uncertainty [Strens, 2000]. Of course this can be very slow, but it is useful to remember that the dynamic programming steps can be computed via simulation over the sampled model, and do not require explicit samples from the system. Convergence of the dynamic

programming inner-loop is improved by keeping maximum likelihood estimates of the value function for each state-action pair. In the bandit case (single-step planning horizon), this method is in fact equivalent to Thompson sampling. Recent work has provided a theoretical characterization of this approach, offering the first Bayesian regret bound for this setting [Osband et al., 2013].

Value of information heuristic

The Bayesian dynamic programming approach does not explicitly take into account the posterior uncertainty when selecting actions, and thus, cannot explicitly select actions which only provide information about the model. In contrast, Dearden et al. [Dearden et al., 1999] proposed to select actions by considering their expected value of information (in addition to their expected reward). Instead of solving the BAMDP directly via Eq. 4.3, the Dirichlet distributions are used to compute a distribution over the state-action values $Q^*(s,a)$, in order to select the action that has the highest expected return and *value of information*. The distribution over Q-values is estimated by sampling MDPs from the posterior Dirichlet distribution, and then solving each sampled MDP (as detailed in §2.2) to obtain different sampled Q-values.

The value of information is used to estimate the expected improvement in policy following an exploration action. Unlike the full Bayes-optimal approach, this is defined myopically, over a 1-step horizon.

Definition 4.3 (Value of perfect information [Dearden et al., 1999])**.** Let $VPI(s,a)$ denote the expected value of perfect information for taking action a in state s. This can be estimated by

$$VPI(s,a) = \int_{-\infty}^{\infty} Gain_{s,a}(x) \Pr(q_{s,a} = x) dx,$$

where

$$Gain_{s,a}(Q^*_{s,a}) = \begin{cases} \mathbf{E}[q_{s,a_2}] - q^*_{s,a} & \text{if } a = a_1 \text{ and } q^*_{s,a} < \mathbf{E}[q_{s,a_2}], \\ q^*_{s,a} - E[q_{s,a_1}] & \text{if } a \neq a_1 \text{ and } q^*_{s,a} > \mathbf{E}[q_{s,a_2}], \\ 0 & \text{otherwise,} \end{cases}$$

assuming a_1 and a_2 denote the actions with the best and second-best

expected values respectively, and $q^*_{s,a}$ denotes a random variable representing a possible value of $Q^*(s,a)$ in some realizable MDP.

The value of perfect information gives an upper bound on the 1-step expected value of exploring with action a. To balance exploration and exploitation, it is necessary to also consider the reward of action a. Thus, under this approach, the goal is to select actions that maximize $\mathbf{E}[q_{s,a}] + VPI(s,a)$.

From a practical perspective, this method is attractive because it can be scaled easily by varying the number of samples. Re-sampling and importance sampling techniques can be used to update the estimated Q-value distribution as the Dirichlet posteriors are updated. The main limitation is that the myopic value of information may provide only a very limited view of the potential information gain of certain actions.

4.5 Online Tree Search Approximation

To select actions using a more complete characterization of the model uncertainty, an alternative is to perform a forward search in the space of hyper-states.

Forward search

An approach of this type was used in the case of a partially observable extension to the BAMDP [Ross et al., 2008a]. The idea here is to consider the current hyper-state and build a fixed-depth forward search tree containing all hyper-states reachable within some fixed finite planning horizon, denoted d. Assuming some default value for the leaves of this tree, denoted $V_0(s,\phi)$, dynamic programming backups can be applied to estimate the expected return of the possible actions at the root hyper-state. The action with highest return over that finite horizon is executed and then the forward search is conducted again on the next hyper-state. This approach is detailed in Algorithm 2 and illustrated in Figure 4.2. The main limitation of this approach is the fact that for most domains, a full forward search (i.e., without pruning of the search tree) can only be achieved over a very short decision horizon, since the number of nodes explored is $\mathcal{O}(|\mathcal{S}|^d)$, where d is the search depth.

Another limitation is the need for a default value function to be used at

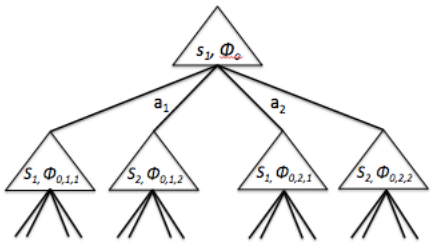

Figure 4.2: An AND-OR tree constructed by the forward search process for the Chain problem. The top node contains the initial state 1 and the prior over the model ϕ_0. After the first action, the agent can end up in either state 1 or state 2 of the Chain and update its posterior accordingly. Note that depending on what action was chosen (1 or 2) and the effect (a or b), different parameters of ϕ are updated as per Algorithm 2

.

the leaf nodes; this can either be a naive estimate, such as the immediate reward, $\max_a R(s, a)$, or a value computed from repeated Bellman backups, such as the one used for the Bayesian dynamic programming approach. The next algorithm we review proposes some solutions to these problems.

We take this opportunity to draw the reader's attention to the survey paper on online POMDP planning algorithms [Ross et al., 2008c], which provides a comprehensive review and empirical evaluation of a range of search-based POMDP solving algorithms, including options for combining offline and online methods in the context of forward search trees. Some of these methods may be much more efficient than those presented above and could be applied to plan in the BAMDP model.

Bayesian Sparse Sampling

Wang et al. [Wang et al., 2005] present an online planning algorithm that estimates the optimal value function of a BAMDP (Equation 4.3) using Monte-Carlo sampling. This algorithm is essentially an adaptation of the Sparse Sampling algorithm [Kearns et al., 1999] to BAMDPs. The original Sparse Sampling approach is very simple: a forward-search tree is grown from the current state to a fixed depth d. At each internal

Algorithm 2 Forward Search Planning in the BAMDP.

function FORWARDSEARCH-BAMDP(s, ϕ, d)
if $d = 0$ **then**
  return $V_0(s, \phi)$
end if
$maxQ \leftarrow -\infty$
for all $a \in \mathcal{A}$ **do**
  $q \leftarrow R(s, a)$
  $\phi'_{s,a} \leftarrow \phi_{s,a}$
  for all $s' \in \mathcal{S}$ **do**
    $\phi'_{s,a,s'} \leftarrow \phi_{s,a,s'} + 1$
    $q \leftarrow q + \gamma \frac{\phi_{s,a,s'}}{\sum_{s'' \in S} \phi_{s,a,s''}}$ FORWARDSEARCH-BAMDP$(s', \phi', d-1)$
    $\phi'_{s,a,s'} \leftarrow \phi_{s,a,s'}$
  end for
  if $q > maxQ$ **then**
    $maxQ \leftarrow q$
  end if
end for
return $maxQ$

node, a fixed number C of next states are sampled from a simulator for each action in $\mathcal{A}$, to create $C|\mathcal{A}|$ children. This is in contrast to the Forward Search approach of Algorithm 2, which considers all possible next states. Values at the leaves are estimated to be zero and the values at the internal nodes are estimated using the Bellman equation based on their children's values. The main feature of Sparse Sampling is that it can be shown to achieve low error with high probability in a number of samples independent of the number of states [Kearns et al., 1999]. A practical limitation of this approach is the exponential dependence on planning horizon.

To extend this to BAMDP, Bayesian Sparse Sampling introduces the following modifications. First, instead of growing the tree evenly by looking at all actions at each level of the tree, the tree is grown stochastically. Actions are sampled according to their likelihood of being optimal, according to their Q-value distributions (as defined by the Dirichlet posteriors); next states are sampled according to the Dirichlet posterior on the model. This is related to Thompson sampling (§3.3),

in that actions are sampled according to their current posterior. The difference is that the Bayesian Sampling explicitly considers the long-term return (as estimated from Monte-Carlo sampling of the posterior over model parameters), whereas Thompson sampling considers just the posterior over immediate rewards. The specific algorithm reported in [Wang et al., 2005] proposes a few improvements on this basic framework: such as, in some cases, sampling from the mean model (rather than the posterior) to reduce the variance of the Q-value estimates in the tree, and using myopic Thompson sampling (rather than the sequential estimate) to decide which branch to sample when growing the tree. This approach requires repeatedly sampling from the posterior to find which action has the highest Q-value at each state node in the tree. This can be very time consuming, and thus, so far the approach has only been applied to small MDPs.

HMDP: A linear program approach for the hyper-state MDP

Castro and Precup [Castro and Precup, 2007] present a similar approach to [Wang et al., 2005]. However, their approach differs on two main points. First, instead of maintaining only the posterior over models, they also maintain Q-value estimates at each state-action pair (for the original MDP states, not the hyper-states) using standard Q-learning [Sutton and Barto, 1998]. Second, values of the hyper-states in the stochastic tree are estimated using *linear programming* [Puterman, 1994] instead of dynamic programming. The advantage of incorporating Q-learning is that it provides estimates for the fringe nodes which can be used as constraints in the LP solution. There is currently no theoretical analysis to accompany this approach. However, the empirical tests on small simulation domains show somewhat better performance than the method of [Wang et al., 2005], but with similar scalability limitations.

Branch and bound search

Branch and bound is a common method for pruning a search tree. The main idea is to maintain a lower bound on the value of each node in the tree (e.g., using partial expansion of the node to some fixed depth with

a lower-bound at the leaf) and then prune nodes whenever they cannot improve the lower-bound (assuming we are maximizing values). In the context of BAMDPs, this can be used to prune hyper-state nodes in the forward search tree [Paquet et al., 2005]. The challenge in this case is to find good bounds; this is especially difficult given the uncertainty over the underlying model. The method has been used in the context of partially observable BAMDP [Png and Pineau, 2011, Png, 2011] using a naive heuristic, $\sum_{d=0}^{D} \gamma^d R_{\max}$, where D is the search depth and $R_{\max}$ is the maximum reward. The method was applied successfully to solve simulated dialogue management problems; computational scalability was achieved via a number of structural constraints, including the parameter tying method proposed by [Poupart et al., 2006].

BOP: Bayesian Optimistic Planning

The BOP method [R. Fonteneau, 2013] is a variant on Branch and bound search in which nodes are expanded based on an upper-bound on their value. The upper-bound at a root node, s_0, is calculated by building an optimistic subtree, $\mathcal{T}^+$, where leaves are expanded using: $x_t = \arg\max_{x \in \mathcal{L}(\mathcal{T}^+)} P(x)\gamma^d$, where $\mathcal{L}(\mathcal{T}^+)$ denotes the fringe of the forward search tree, $P(x)$ denotes the probability of reaching node x and d denotes the depth of node x.

Theoretical analysis shows that the regret of BOP decreases exponentially with the planning budget. Empirical results on the classic 5-state Chain domain (Figure 4.1) confirm that the performance improves with the number of nodes explored in the forward search tree.

BAMCP: Bayes-adaptive Monte-Carlo planning

In recent years, Monte-Carlo tree search methods have been applied with significant success to planning problems in games and other domains [Gelly et al., 2012]. The BAMCP approach exploits insights from this family of approaches to tackle the problem of joint learning and planning in the BAMDP [Guez et al., 2012].

BAMCP provides a Bayesian adaptation of an algorithm, called *Upper Confidence bounds applied to Trees* (UCT) [Kocsis and Szepesvari, 2006]. This approach is interesting in that it incorporates two different

policies (UCT and rollout) to traverse and grow the forward search tree. Starting at the root node, for any node that has been previously visited, it uses the UCT criteria

$$a^* = \arg\max_a Q(s,h,a) + c\sqrt{\frac{\log\left(n(s,h)\right)}{n(s,h,a)}}$$

to select actions. Along the way, it updates the statistics $n(s,h)$ (the number of times the node corresponding to state s and history h has been visited in the search tree) and $n(s,h,a)$ (the number of times action a was chosen in this node); these are initialized at 0 when the tree is created. Once it reaches a leaf node, instead of using a default value function (or bound), it first selects any untried action (updating its count to 1) and then continues to search forward using a *rollout* policy until it reaches a given depth (or terminal node). The nodes visited by the rollout policy are not added to the tree (i.e., no $n(\cdot)$ statistics are preserved).

To apply this to the Bayesian context, BAMCP must select actions according to the posterior of the model parameters. Rather than sampling multiple models from the posterior, BAMCP samples a single model P_t at the root of the tree and uses this same model (without posterior updating) throughout the search tree to sample next states, after both UCT and rollout actions. In practice, to reduce computation in large domains, the model P_t is not sampled in its entirety at the beginning of the tree building process, rather, it is generated lazily as samples are required.

In addition, to further improve efficiency, BAMCP uses learning within the forward rollouts to direct resources to important areas of the search space. Rather than using a random policy for the rollouts, a model-free estimate of the value function is maintained

$$\hat{Q}(s_t,a_t) \leftarrow \hat{Q}(s_t,a_t) + \alpha\big(r_t + \gamma \max_a \hat{Q}(s_{t+1},a) - \hat{Q}(s_t,a_t)\big),$$

and actions during rollouts are selected according to the ϵ-greedy policy defined by this estimated $\hat{Q}$ function.

BAMCP is shown to converge (in probability) to the optimal *Bayesian* policy (denoted $V^*(s,h)$ in general, or $V^*(s,\phi)$ when the posterior is constrained to a Dirichlet distribution). The main complexity

result for BAMCP is based on the UCT analysis and shows that the error in estimating $V(s_t, h_t)$ decreases as $\mathcal{O}\Big(\log\left(n(s_t, h_t)\right)/n(s_t, h_t)\Big)$.

Empirical evaluation of BAMCP with a number of simulation domains has shown that it outperforms Bayesian Dynamic Programming, the Value of Information heuristic, BFS3 [Asmuth and Littman, 2011], as well as BEB [Kolter and Ng, 2009] and SmartSampler [Castro and Precup, 2010], both of which will be described in the next section. A good part of this success could be attributed to the fact that unlike many forward search sparse sampling algorithm (e.g., BFS3), BAMCP can take advantage of its learning during rollouts to effectively bias the search tree towards good solutions.

4.6 Methods with Exploration Bonus to Achieve PAC Guarantees

We now review a class of algorithms for the BAMDP model that are guaranteed to select actions such as to incur only a small loss compared to the optimal Bayesian policy. Algorithmically, these approaches are similar to those examined in §4.5 and typically require forward sampling of the model and decision space. Analytically, these approaches have been shown to achieve bounded error in a polynomial number of steps using analysis techniques from the *Probably Approximately Correct* (PAC) literature. These methods are rooted in earlier papers showing that reinforcement learning in finite MDPs can be achieved in a polynomial number of steps [Kearns and Singh, 1998, Brafman and Tennenholtz, 2003, Strehl and Littman, 2005]. These earlier papers did not assume a Bayesian learning framework; the extension to Bayesian learning was first established in the BOSS (Best of Sampled Sets) approach.

The main idea behind many of the methods presented in this section is the notion of *Optimism in the Face of Uncertainty*, which suggests that, when in doubt, an agent should act according to an optimistic model of the MDP; in the case where the optimistic model is correct, the agent will gather reward, and if not, the agent will receive valuable information from which it can infer a better model.

BFS3: Bayesian Forward Search Sparse Sampling

The BFS3 approach [Asmuth and Littman, 2011] is an extension to the Bayesian context of the Forward Search Sparse Sampling (FSSS) approach [Walsh et al., 2010]. The Forward Search Sparse Sampling itself extends the Sparse Sampling approach [Kearns et al., 1999] described above in a few directions. In particular, it maintains both lower and upper bounds on the value of each state-action pair, and uses this information to direct forward rollouts in the search tree. Consider a node s in the tree, then the next action is chosen greedily with respect to the upper-bound $U(s, a)$. The next state s' is selected to be the one with the largest difference between its lower and upper bound (weighted by the number of times it was visited, denoted by $n(s, a, s')$), i.e., $s' \leftarrow \arg\max_{s' \sim P(\cdot|s,a)} \left(U(s') - L(s')\right) n(s, a, s')$. This is repeated until the search tree reaches the desired depth.

The BFS3 approach takes this one step further by building the sparse search tree over hyper-states (s, ϕ), instead of over simple states s. As with most of the other approaches presented in this section, the framework can handle model uncertainty over either the transition parameters and/or the reward parameters.

Under certain conditions, one can show that BFS3 chooses at most a polynomial number of sub-optimal actions compared to an policy.

Theorem 4.1. [Asmuth, 2013][1] With probability at least $1 - \delta$, the expected number of sub-ϵ-Bayes-optimal actions taken by BFS3 is at most $BSA(S + 1)d/\delta_t$ under assumptions on the accuracy of the prior and optimism of the underlying FSSS procedure.

Empirical results show that the method can be used to solve small domains (e.g., 5x5 grid) somewhat more effectively than a non-Bayesian method such as RMAX [Brafman and Tennenholtz, 2003]. Results also show that BFS3 can take advantage of structural assumptions in the prior (e.g., parameter tying) to tackle much larger domains, up to 10^{16} states.

[1] We have slightly modified the description of this theorem, and others below, to improve legibility of the paper. Refer to the original publication for full details.

BOSS: Best of Sample Sets

As per other model-based Bayesian RL approaches presented above, BOSS [Asmuth et al., 2009] maintains uncertainty over the model parameters using a parametric posterior distribution and incorporates new information by updating the posterior. In discrete domains, this posterior is also typically represented using Dirichlet distributions.

When an action is required, BOSS draws a set of sample models $\mathcal{M}_i, i = 1, \ldots, K$, from the posterior ϕ_t. It then creates a hyper-model $\mathcal{M}\#$, which has the same state space $\mathcal{S}$ as the original MDP, but has $K|\mathcal{A}|$ actions, where a_{ij} corresponds to taking action $a_j \in \mathcal{A}$ in model $\mathcal{M}_i$. The transition function for action $a_{i\cdot}$ is constructed directly by taking the sampled transitions P_i from $\mathcal{M}_i$. It then solves the hyper-model $\mathcal{M}\#$ (e.g., using dynamic programming methods) and selects the best action according to this hyper-model. This sampling procedure is repeated a number of times (as determined by B, the *knownness* parameter) for each state-action pair, after which the policy of the last $\mathcal{M}\#$ is retained.

While the algorithm is simple, the more interesting contribution is in the analysis. The main theoretical result shows that assuming a certain knownness parameter B, the value at state s when visited by algorithm $\mathfrak{A}$ at time t (which we denote $V^{\mathfrak{A}_t}(s_t)$), is very close to the optimal value of the correct model (denoted V^*) with high probability in all but a small number of steps. For the specific case of Dirichlet priors on the model, it can be shown that the number of necessary samples, f, is a polynomial function of the relevant parameters.

Theorem 4.2. [Asmuth et al., 2009] When the knownness parameter $B = \max_{s,a} f(s, a, \epsilon(1-\gamma)^2, \frac{\delta}{|\mathcal{S}||\mathcal{A}|}, \frac{\delta}{|\mathcal{S}|^2|\mathcal{A}|^2K})$, then with probability at least $1 - 4\delta$, $V^{\mathfrak{A}_t}(s_t) \geq V^*(s_t) - 4\epsilon$ in all but $\xi(\epsilon, \delta) = \mathcal{O}\left(\frac{|\mathcal{S}||\mathcal{A}|B}{\epsilon(1-\gamma)^2} \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)}\right)$ steps.

Experimental results show that empirically, BOSS performs similarly to BEETLE and Bayesian dynamic programming in simple problems, such as the Chain problem (Figure 4.1). BOSS can also be extended to take advantage of structural constraints on the state relations to tackle larger problems, up to a 6×6 maze.

Castro and Precup [Castro and Precup, 2010] proposed an improvement on BOSS, called SmartSampler, which addresses the problem of how many models to sample. The main insight is that the number of sampled models should depend on the variance of the posterior distribution. When the variance is larger, more models are necessary to achieve good value function estimation. When the variance is reduced, it is sufficient to sample a smaller number of models. Empirical results show that this leads to reduced computation time and increased accumulated reward, compared to the original BOSS.

BEB: Bayesian Exploration Bonus

Bayesian Exploration Bonus (BEB) is another simple approach for achieving Bayesian reinforcement learning with guaranteed bounded error within a small number of samples [Kolter and Ng, 2009]. The algorithm simply chooses actions greedily with respect to the following value function:

$$\tilde{V}_t^*(s,\phi) = \max_{a\in\mathcal{A}} \left\{ R(s,a) + \frac{\beta}{1+\sum_{s'\in\mathcal{S}} \phi_{s,a,s'}} + \sum_{s'\in\mathcal{S}} P(s'|s,\phi,a)\tilde{V}_{t-t}^*(s',\phi) \right\}$$

Here the middle term on the right-hand side acts as exploration bonus, whose magnitude is controlled by parameter β. It is worth noting that the posterior (denoted by ϕ) is not updated in this equation, and thus, the value function can be estimated via standard dynamic programming over the state space (similar to BOSS and Bayesian dynamic programming).

The exploration bonus in BEB decays with $1/n$ (consider $n \sim \sum_{s'\in\mathcal{S}} \phi_{s,a,s'}$), which is significantly faster than similar exploration bonuses in the PAC-MDP literature, for example the MBIE-EB algorithm [Strehl and Littman, 2008], which typically declines with $1/\sqrt{n}$. It is possible to use this faster decay because BEB aims to match the performance of the optimal *Bayesian* solution (denoted $V^*(s,\phi)$), as defined in Theorem 4.3. We call this property PAC-BAMDP. This is in contrast to BOSS where the analysis is with respect to the optimal solution of the correct model (denoted $V^*(s)$).

Theorem 4.3. [Kolter and Ng, 2009] Let $\mathcal{A}_t$ denote the policy followed

by the BEB algorithm (with $\beta = 2H^2$) at time t, and let s_t and ϕ_t be the corresponding state and posterior belief. Also suppose we stop updating the posterior for a state-action pair when $\sum_{s' \in \mathcal{S}} \phi_{s,a,s'} \geq 4H^3/\epsilon$. Then with probability at least $1 - \delta$, for a planning horizon of H, we have

$$V^{\mathcal{A}_t}(s_t, \phi_t) \geq V^*(s_t, \phi_t) - \epsilon$$

for all but m time steps, where

$$m = \mathcal{O}\left(\frac{|\mathcal{S}||\mathcal{A}|H^6}{\epsilon^2} \log \frac{|\mathcal{S}||\mathcal{A}|}{\delta}\right).$$

It is worth emphasizing that because BEB considers optimality with respect to the value of the Bayesian optimal policy, it is possible to obtain theoretical results that are tighter (i.e., fewer number of steps) with respect to the optimal value function. But this comes at a cost, and in particular, there are some MDPs for which the BEB algorithm, though it performs closely to the Bayesian optimum, $V^*(s, \phi)$, may never find the actual optimal policy, $V^*(s)$. This is illustrated by an example in the dissertation of Li [Li, 2009] (see Example 9, p.80), and formalized in Theorem 4.4 (here $n(s, a)$ denotes the number of times the state-action pair s, a has been observed).

Theorem 4.4. [Kolter and Ng, 2009] Let $\mathcal{A}_t$ denote the policy followed by an algorithm using any (arbitrary complex) exploration bonus that is upper bounded by $\frac{\beta}{n(s,a)^p}$, for some constants β and $p > 1/2$. Then there exists some MDP, $\mathcal{M}$, and parameter, $\epsilon_0(\beta, p)$, such that with probability greater than $\delta_0 = 0.15$,

$$V^{\mathcal{A}_t}(s_t) < V^*(s_t) - \epsilon_0,$$

will hold for an unbounded number of time steps.

Empirical evaluation of BEB showed that in the Chain problem (Figure 4.1), it could outperform a (non-Bayesian) PAC-MDP algorithm in terms of sample efficiency and find the correct optimal policy.

VBRB: Variance-Based Reward Bonus

The *Variance-based reward* approach also tackles the problem of Bayesian RL by applying an exploration bonus. However in this case,

the exploration bonus depends on the variance of the model parameters with respect to the posterior [Sorg et al., 2010], i.e.,

$$\tilde{V}_t^*(s,\phi) = \max_{a\in\mathcal{A}} \left\{ R(s,\phi,a) + \hat{R}_{s,\phi,a} + \sum_{s'\in\mathcal{S}} P(s'|s,\phi,a)\tilde{V}_{t-t}^*(s',\phi) \right\},$$

where the reward bonus $\hat{R}_{s,\phi,a}$ is defined as

$$\beta_R\ \sigma_{R(s,\phi,a)} + \beta_P \sqrt{\sum_{s'\in\mathcal{S}} \sigma^2_{P(s'|s,\phi,a)}}\ ,$$

with

$$\sigma^2_{R(s,\phi,a)} = \int_\theta R(s,\theta,a)^2 b(\theta) d\theta - R(s,\phi,a)^2, \tag{4.4}$$

$$\sigma^2_{P(s'|s,\phi,a)} = \int_\theta P(s'|s,\theta,a)^2 b(\theta) d\theta - P(s'|s,\phi,a)^2, \tag{4.5}$$

and β_R and β_P are constants controlling the magnitude of the exploration bonus.[2]

The main motivation for considering a variance-based bonus is that the error of the algorithm can then be analyzed by drawing on Chebyshev's inequality (which states that with high probability, the deviation of a random variable from its mean is bounded by a multiple of its variance). The main theoretical result concerning the variance-based reward approach bounds the sample complexity with respect to the optimal policy of the true underlying MDP, like BOSS (and unlike BEB).

Theorem 4.5. [Sorg et al., 2010] Let the sample complexity of state s and action a be $C(s,a) = f(b_0, s, a, \frac{1}{4}\epsilon(1-\gamma)^2, \frac{\delta}{|\mathcal{S}||\mathcal{A}|}, \frac{\delta}{2|\mathcal{S}|^2|\mathcal{A}|^2})$. Let the internal reward be defined as $\hat{R}(s,\phi,a) = \frac{1}{\sqrt{\rho}}\left(\sigma_{R(s,\phi,a)} + \frac{\gamma|\mathcal{S}|}{1-\gamma}\sqrt{\sum_{s'}\sigma^2_{P(s'|s,\phi,a)}}\right)$ with $\rho = \frac{\delta}{2|\mathcal{S}|^2|\mathcal{A}|^2}$. Let θ^* be the random true model parameters distributed according to the prior belief ϕ_0. The variance-based reward algorithm will follow a 4ϵ-optimal

[2] The approach is presented here in the context of Dirichlet distributions. There are ways to generalize this use of variance of the reward as an exploration bonus for other Bayesian priors [Sorg et al., 2010].

policy from its current state, with respect to the MDP θ^*, on all but $\mathcal{O}\left(\frac{\sum_{s,a} C(s,a)}{\epsilon(1-\gamma)^2} \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)}\right)$ time steps with probability at least $1-4\delta$.

For the case where uncertainty over the transition model is modelled using independent Dirichlet priors (as we have considered throughout this section), the sample complexity of this approach decreases as a function of $\mathcal{O}(1/\sqrt{n})$. This is consistent with other PAC-MDP results, which also bound the sample complexity to achieve small error with respect to the optimal policy of the true underlying MDP. However, it is not as fast as the BEB approach, which bounds error with respect to the best Bayesian policy.

Empirical results for the variance-based approach show that it is competitive with BEETLE, BOSS and BEB on versions of the Chain problem that use parameter tying to reduce the space of model uncertainty, and shows better performance for the variant of the domain where uncertainty over all state-action pairs is modelled with an independent Dirichlet distribution. Results also show superior performance on a 4×4 grid-world inspired by the Wumpus domain of [Russell and Norvig, 2002].

BOLT: Bayesian Optimistic Local Transitions

Both BEB and the variance-based reward approach encourage exploration by putting an exploration bonus on the reward, and solving this modified MDP. The main insight in BOLT is to put a similar exploration bonus on the transition probabilities [Araya-Lopez et al., 2012]. The algorithm is simple and modelled on BOSS. An extended action space is considered: $\mathcal{A}' = \mathcal{A} \times S$, where each action in $\zeta = (a, \sigma) \in \mathcal{A}'$ has transition probability $\hat{P}(s'|s,\zeta) = \mathbf{E}[P(s'|s,a)|s,\phi,\lambda^{\eta}_{s,a,\sigma}]$, with ϕ being the posterior on the model parameters and λ^{η} being a set of η *imagined* transitions $\lambda^{\eta}_{s,a,\sigma} = \{(s,a,\sigma),\ldots,(s,a,\sigma)\}$. The extended MDP is solved using standard dynamic programming methods over a horizon H, where H is a parameter of BOLT, not the horizon of the original problem. The effect of this exploration bonus is to consider an action ζ that takes the agent to state $\sigma \in \mathcal{S}$ with greater probability than has been observed in the data, and by optimizing a pol-

icy over all such possible actions, we consider an optimistic evaluation over the possible set of models. The parameter η controls the extent of the optimistic evaluation, as well as its computational cost (larger η means more actions to consider). When the posterior is represented using standard Dirichlet posteriors over model parameters, as considered throughout this section, it can be shown that BOLT is always optimistic with respect to the optimal Bayesian policy [Araya-Lopez et al., 2012].

Theorem 4.6. [Araya-Lopez et al., 2012] Let $\mathcal{A}_t$ denote the policy followed by BOLT at time t with $\eta = H$. Let also s_t and ϕ_t be the corresponding state and posterior at the time. Then, with probability at least $1 - \delta$, BOLT is ϵ-close to the optimal Bayesian policy

$$V^{\mathcal{A}_t}(s_t, \phi_t) < V^*(s_t, \phi_t) - \epsilon,$$

for all but $\widetilde{\mathcal{O}}(\frac{|\mathcal{S}||\mathcal{A}|\eta^2}{\epsilon^2(1-\gamma)^2}) = \widetilde{\mathcal{O}}(\frac{|\mathcal{S}||\mathcal{A}|H^2}{\epsilon^2(1-\gamma)^2})$ time steps.

To simplify, the parameter H can be shown to depend on the desired correctness (ϵ, δ).

Empirical evaluation of BOLT on the Chain problem shows that while it may be outperformed by BEB with well-tuned parameters, it is more robust to the choice of parameter (H for BOLT, β for BEB). The authors suggest that BOLT is more stable with respect to the choice of parameter because optimism in the transitions is bounded by laws of probability (i.e., even with a large exploration bonus, η, the effect of extended actions ζ will simply saturate), which is not the case when the exploration bonus is placed on the rewards.

BOLT was further extended, in the form of an algorithm called POT: Probably Optimistic Transition [Kawaguchi and Araya-Lopez, 2013], where the parameter controlling the extended bonus, η, is no longer constant. Instead, the transition bonus is estimated online for each state-action pair using a UCB-like criteria that incorporates the notion of the current estimate of the transition and the variance of that estimate (controlled by a new parameter β). One advantage of the method is that empirical results are improved (compared to BOLT, BEB, and VBRB) for domains where the prior is good, though empirical performance can suffer when the prior is misspecified. POT is also

shown to have tighter bounds on the sample complexity. However, the analysis is done with respect to a modified optimality criteria, called "Probably Upper Bounded Belief-based Bayesian Planning", which is weaker than the standard Bayesian optimality, as defined in §4.2.

Table 4.1 summarizes the key features of the online Bayesian RL methods presented in this section. This can be used as a quick reference to visualize similarities and differences between the large number of related approaches. It could also be used to devise new approaches, by exploring novel combinations of the existing components.

4.7 Extensions to Unknown Rewards

Most work on model-based Bayesian RL focuses on unknown transition functions and assumes the reward function is known. This is seen as the more interesting case because there are many domains in which dynamics are difficult to model *a priori*, whereas the reward function is set by domain knowledge. Nonetheless a number of papers explicitly consider the case where the reward function is uncertain [Dearden et al., 1999, Strens, 2000, Wang et al., 2005, Castro and Precup, 2007, Sorg et al., 2010]. The BAMDP model extends readily to this case, by extending the set of hyper-states to full set of unknown parameters: $\mathcal{S}' \in \mathcal{S} \times \Theta$, with $\theta = (\phi, \vartheta)$, with ϑ representing the prior over the unknown reward function. The next challenge is to choose an appropriate distribution for this prior.

The simplest model is to assume that each state-action pair generates a binary reward, $\vartheta = \{0, 1\}$, according to a Bernoulli distribution [Wang et al., 2005, Castro and Precup, 2007]. In this case, a conjugate prior over the reward can be captured with a Beta distribution (see §2.5.1). This is by far the most common approach when dealing with bandit problems. In the cases where the reward function is drawn from a set of discrete values, $\vartheta = \{r_1, r_2, \ldots, r_k\}$, it can be modeled by a Dirichlet prior over those values. However, many MDP domains are characterized by more complex (e.g., real-valued) reward functions.

In those cases, an alternative is to assume that the reward is

Algorithm	Depth	Actions selected in search	Next-states considered	Posterior sampling	Solution method	Theoretical properties
Bayesian DP	N/A	$\forall a \in \mathcal{A}$	$\forall s' \in S$	Sample 1 per step	$Q()$ solved by DP	MDP opt. $\rightarrow \infty$ Bounded expected regret
Value of information	N/A	Information gain	$\forall s' \in S$	Sample K per step	$Q()$ solved by DP	Not known
Forward search	fixed d	$\forall a \in \mathcal{A}$	$\forall s' \in S$	Re-sample at node	Backups in tree, heuristic at leave	Bayes-opt. $\rightarrow \infty$
Bayesian Sparse Sampling	variable d	$a \sim E[\hat{R}(s,a)]$	$s' \sim \hat{Pr}(s'\|s,a,\phi)$	Re-sample at node	Backups in tree, heuristic at leaves	Not known
HMDP	fixed d	$a \sim rand()$	$s' \sim \hat{Pr}(s'\|s,a,\phi)$	Re-sample at node	Backups in tree, $Q()$ by LP at leaves	Not known
Branch and Bound	variable d	$\forall a \in \mathcal{A}$, except if $\exists a' U(s,\phi,a) < L(s,\phi,a')$	$\forall s' \in S$	Re-sample at node	Backups in tree, heuristic at leaves	Bayes-opt. $\rightarrow \infty$
BOP	variable d	$\arg\max_a B(x,a)$	$\forall s' \in S$	Re-sample at node	Backups in tree, $\frac{1}{1-\gamma}$ at leaves	Bayes-opt. $\rightarrow \infty$
BAMCP	fixed d	UCT criteria	$s' \sim \hat{Pr}(s'\|s,a,\phi)$	Sample 1 per step	Rollouts with fixed policy $\hat{\mu}^*$	Bayes-opt. $\rightarrow \infty$

Table 4.1: Online Model-based Bayesian RL methods (DP=Dynamic programming, LP = Linear programming, U=Upper-bound, L=Lower-bound)

Algorithm	Depth	Actions selected in search	Next-states considered	Posterior sampling	Solution method	Theoretical properties
BFS3	variable d	$\arg\max_a U(s,\phi,a)$	$\arg\max_{s'}(U(s') - L(s'))$	Re-sample at node	Backups in tree, heuristic at leaves	PAC guarantee
BOSS	N/A	$\mathcal{A} \times K$	$\forall s' \in S$	Sample K per step	Q() solved by DP	PAC-MDP
Smart-Sampler	N/A	$\mathcal{A} \times K$	$\forall s' \in S$	Sample K per step, $K = \frac{\max_{s'} var(P(s,a,s'))}{\epsilon}$	Q() solved by DP	PAC-MDP
BEB	N/A	$\forall a \in \mathcal{A}$	$\forall s' \in S$	N/A	$Q()$ + reward bonus solved by DP	PAC-BAMDP
VBRB rewards	N/A	$\forall a \in \mathcal{A}$	$\forall s' \in S$	N/A	$Q()$ + variance bonus solved by DP	PAC-MDP
BOLT	N/A	$\mathcal{A} \times \mathcal{S}$	$\forall s' \in S$	Create η per step	$Q()$ solved by DP	PAC-BAMDP

Table 4.1: (cont'd) Online Model-based Bayesian RL methods (DP=Dynamic programming, LP = Linear programming, U=Upper-bound, L=Lower-bound)

Gaussian distributed, i.e., $\vartheta = \{\mu, \sigma\}$. In this case the choice of prior on the standard deviation σ is of particular importance; a uniform prior could lead the posterior to converge to $\sigma \to 0$. Following Sterns [Strens, 2000], we can define the precision $\psi = 1/\sigma$ with prior density $f(\psi) \propto \psi \exp(-\psi^2\sigma_0^2/2)$; this will have a maximum at $\sigma = \sigma_0$. We can also define the prior on the mean to be $f(\mu) \propto \mathcal{N}(\mu_0, \sigma^2)$. The parameters (μ_0, σ_0) capture any prior knowledge about the mean and standard deviation of the reward function. After n observations of the reward with sample mean $\hat{\mu}$ and sample variance $\hat{\sigma}$, the posterior distribution on ψ is defined by: $f(\psi) \propto \psi^{n-1} \exp\big(-\psi^2(n\hat{\sigma} + \sigma_0^2)/2\big)$, and the posterior on μ is defined by $f(\mu) \propto \mathcal{N}(\hat{\mu}, \sigma^2/n)$, where $\sigma = 1/\psi$. Note that the posterior on the variance is updated first and used to calculate the posterior on the mean.

Most of the online model-based BRL algorithms presented in Table 4.1 extend readily to the case of unknown rewards, with the added requirement of sampling the posterior over rewards (along with the posterior over transition functions). For an algorithm such as BEB, that uses an exploration bonus in the value function, it is also necessary to incorporate the uncertainty over rewards within that exploration bonus. Sorg et al, [Sorg et al., 2010] deals with the issue by expressing the bonus over the variance of the unknown parameters (including the unknown rewards), as in Eq. 4.5.

4.8 Extensions to Continuous MDPs

Most of the work reviewed in this section focuses on discrete domains. In contrast, many of the model-free BRL methods concern the case of continuous domains, where the Bayesian approach is used to infer a posterior distribution over value functions, conditioned on the state-action-reward trajectory observed in the past.

The problem of optimal control under uncertain model parameters for continuous systems was originally introduced by Feldbaum [1961], as the theory of dual control (also referred to as adaptive control or adaptive dual control). Several authors studied the problem for different classes of time-varying systems [Filatov and Unbehauen, 2000,

Wittenmark, 1995]: linear time invariant systems under partial observability [Rusnak, 1995], linear time varying Gaussian models with partial observability [Ravikanth et al., 1992], nonlinear systems with full observability [Zane, 1992], and nonlinear systems with partial observability [Greenfield and Brockwell, 2003, Ross et al., 2008b].

This last case is closest mathematically to the Bayes-Adaptive MDP model considered throughout this paper. The dynamical system is described by a Gaussian transition model (not necessarily linear):

$$s_t = g_T(s_{t-1}, a_{t-1}, V_t),$$

where g_T is a specified function, and $V_t \sim \mathcal{N}(\mu_v, \Sigma_v)$ is an unknown k-variate normal distribution with mean vector μ_v and covariance matrix Σ_v. Here the prior distribution on V_t is represented using a Normal-Wishart distribution [Ross et al., 2008b]. Particle filters using Monte-Carlo sampling methods are used to track the posterior over the parameters of this distribution. Planning is achieved using a forward search algorithm similar to Algorithm 2.

The Bayesian DP strategy (§4.4) and its analysis were also extended to the continuous state and action space and average-cost setting, under the assumption of smoothly parameterizable dynamics [Abbasi-Yadkori and Szepesvari, 2015]. The main algorithmic modification in this extension is to incorporate a specific schedule for updating the policy, that is based on a measure of uncertainty.

Another approach proposed by Dallaire et al. [2009] allows flexibility over the choice of transition function. Here the transition and reward functions are defined by:

$$s_t = g_T(s_{t-1}, a_{t-1}) + \epsilon_T, \tag{4.6}$$

$$r_t = g_R(s_t, a_t) + \epsilon_R, \tag{4.7}$$

where g_T and g_R are modelled by (independent) Gaussian Processes (as defined in §2.5.2) and ϵ_T and ϵ_R are zero-mean Gaussian noise terms. Belief tracking is done by updating the Gaussian process. Planning is achieved by a forward search tree similar to Algorithm 2.

It is also worth pointing out that the Bayesian Sparse Sampling approach [Wang et al., 2005] has also been extended to the case where

the reward function is expressed over a continuous action space and represented by a Gaussian process. In this particular case, the authors only considered Bayesian inference of the reward function for a single state and a single-step horizon problem (i.e., a bandit with continuous actions). Under these conditions, inferring the reward function is the same as inferring the value function, so no planning is required.

4.9 Extensions to Partially Observable MDPs

We can extend the BAMDP model to capture uncertainty over the parameters of a POMDP (as defined in §2.3) by introducing the Bayes-Adaptive POMDP (BAPOMDP) model [Ross et al., 2008a, 2011]. Given a POMDP model, $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P, \Omega, P_0, R \rangle$, we define a corresponding BAPOMDP as follows:

Model 7 (Bayes-Adaptive POMDP) Define a BAPOMDP $\mathcal{M}$ to be a tuple $\langle \mathcal{S}', \mathcal{A}, P', P'_0, R' \rangle$ where
• $\mathcal{S}'$ is the set of hyper-states, $\mathcal{S} \times \Phi \times \Psi$,
• $\mathcal{A}$ is the set of actions,
• $P'(\cdot|s, \phi, \psi, a)$ is the transition function between hyper-states, conditioned on action a being taken in hyper-state (s, ϕ, ψ),
• $\Omega'(\cdot|s, \phi, \psi, a)$ is the observation function, conditioned on action a being taken in hyper-state (s, ϕ, ψ),
• $P'_0 \in \mathcal{P}(\mathcal{S} \times \Phi \times \Psi)$ combines the initial distribution over physical states, with the prior over transition functions ϕ_0 and the prior over observation functions ψ_0,
• $R'(s, \phi, \psi, a) = R(s, a)$ represents the reward obtained when action a is taken in state s.

Here Φ and Ψ capture the space of Dirichlet parameters for the conjugate prior over the transition and observation functions, respectively. The Bayesian approach to learning the transition P and observation Ω involves starting with a prior distribution, which we denote ϕ_0 and ψ_0, and maintaining the posterior distribution over ϕ and ψ after observing the history of action-observation-rewards. The belief can be tracked using a Bayesian filter, with appropriate handing of the observations.

Using standard laws of probability and independence assumptions, we have

$$\begin{aligned}&\mathcal{P}(s', \phi', \psi', z|s, \phi, \psi, a) = \\ &\mathcal{P}(s'|s, a, \phi)\mathcal{P}(o|a, s', \psi)\mathcal{P}(\phi'|\phi, s, a, s')\mathcal{P}(\psi'|\psi, a, s', o).\end{aligned} \tag{4.8}$$

As in the BAMDP case (Eq. 4.2), $\mathcal{P}(s'|s, a, \phi)$ corresponds to the expected transition model under the Dirichlet posterior defined by ϕ, and $\mathcal{P}(\phi'|\phi, s, a, s')$ is either 0 or 1, depending on whether ϕ' corresponds to the posterior after observing transition (s, a, s') from prior ϕ. Hence $\mathcal{P}(s'|s, a, \phi) = \frac{\phi_{s,a,s'}}{\sum_{s'' \in \mathcal{S}} \phi_{s,a,s''}}$ and $\mathcal{P}(\phi'|\phi, s, a, s') = \mathbb{I}(\phi'_{s,a,s'} = \phi_{s,a,s'} + 1)$. Similarly, $\mathcal{P}(o|a, x', \psi) = \frac{\psi_{s',a,o}}{\sum_{o' \in \mathcal{O}} \psi_{s',a,o}}$ and $\mathcal{P}(\psi'|\psi, s', a, o) = I(\psi'_{s',a,o} = \psi_{s',a,o} + 1)$.

Also as in the BAMDP, the number of possible hyper-states, (s, ϕ, ψ), grows exponentially (by a factor of $|\mathcal{S}|$) with the prediction horizon. What is particular to the BAPOMDP is that the number of possible beliefs for a *given trajectory*, $(a_0, o_1, \ldots, a_t, o_{t+1})$, also grows exponentially. In contrast, given an observed trajectory, the belief in the BAMDP corresponds to a single hyper-state. In the BAPOMDP, value-based planning requires estimating the Bellman equation over all possible hyper-states for every belief:

$$\begin{aligned}V^*\big(b_t(s, \phi, \psi)\big) = \max_a \Big\{ &\sum_{s,\phi,\psi} R'(s, \phi, \psi, a) b_t(s, \phi, \psi) \\ &+ \gamma \sum_{o \in \mathcal{O}} \mathcal{P}(o|b_{t-1}, a) V^*\big(\tau(b_t, a, o)\big) \Big\}.\end{aligned} \tag{4.9}$$

This is intractable for most problems, due to the large number of possible beliefs. It has been shown that Eq. 4.9 can be approximated with an ϵ-optimal value function defined over a smaller finite-dimensional space [Ross et al., 2011]. In particular, there exists a point where if we simply stop incrementing the counts (ϕ, ψ), the value function of that approximate BAPOMDP (where the counts are bounded) approximates the BAPOMDP within some $\epsilon > 0$. In practice, that space is still very large.

The Minerva approach [Jaulmes et al., 2005] overcomes this problem by assuming that there is an oracle who is willing to provide full state

identification at any time step. This oracle can be used to deterministically update the counts ψ and ϕ (rather than keeping a probability distribution over them, as required when the state is not fully observable). It is then possible to sample a fixed number of models from the Dirichlet posterior, solve these models using standard POMDP planning techniques, and sample an action from the solved models according to the posterior weight over the corresponding model.

The assumption of a state oracle in Minerva is unrealistic for many domains. The approach of Atrash and Pineau [2009] weakens the assumption to an action-based oracle, which can be called on to reveal the optimal POMDP action at any time step for the current belief over states. This oracle does not consider the uncertainty over model parameters and computes its policy based on the correct parameters. There are some heuristics to decide when to query the oracle, but this is largely an unexplored question.

An alternative approach for the BAPOMDP is to use the Bayes risk as a criterion for choosing actions [Doshi et al., 2008, Doshi-Velez et al., 2011]. This framework considers an extended action set, which augments the initial action set with a set of query actions corresponding to a consultation with an oracle. Unlike the oracular queries used in Atrash and Pineau [2009], which explicitly ask for the optimal action (given the current belief), the oracular queries in Doshi et al. [2008] request confirmation of an optimal action choice: *I think a_i is the best action. Should I do a_i?* If the oracle answers to the negative, there is a follow-up query: *Then I think a_j is best. Is that correct?*, and so on until a positive answer is received. In this framework, the goal is to select actions with smallest expected loss, defined here as the Bayes risk (cf. the regret definition in Chapter 2.1):

$$BR(a) = \sum_{s,\phi,\psi} Q(b_t(s,\phi,\psi,a))b_t(s,\phi,\psi) - \sum_{s,\phi,\psi} Q(b_t(s,\phi,\psi,a^*))b_t(s,\phi,\psi),$$

where $Q(\cdot)$ is just Eq. 4.9 without the maximization over actions and a^* is the optimal action at $b_t(s,\phi,\psi)$. Because the second term is independent of the action choice, to minimize the Bayes risk, we simply consider maximizing the first term over actions. The analysis of this algorithm has yielded a bound on the number of samples needed to

ensure ϵ-error. The bound is quite loose in practice, but at least provides some upper-bound on the number of queries needed to achieve good performance. Note that this analysis is based on the Bayes risk criterion that provides a myopic view of uncertainty (i.e., it assumes that the next action will resolve the uncertainty over models).

The planning approach suggested by Ross et al. [2011] aims to approximate the optimal BAPOMDP strategy by employing a forward search similar to that outlined in Algorithm 2. In related work, Png and Pineau [2011] use a branch-and-bound algorithm to approximate the BAPOMDP solution. Many of the other techniques outlined in Table 4.1 could also be extended to the BAPOMDP model.

Finally, it is worth mentioning that the method of Dallaire et al. [2009], described in §4.8, is also able to handle continuous partially observable domains by using an additional Gaussian process for the observation function.

4.10 Extensions to Other Priors and Structured MDPs

The work and methods presented above focus on the case where the model representation consists of a discrete (flat) set of states. For many larger domains, it is common to assume that the states are arranged in a more sophisticated structure, whether a simple clustering or more complicated hierarchy. It is therefore interesting to consider how Bayesian reinforcement learning methods can be used in those cases.

The simplest case, called *parameter tying* in previous sections, corresponds to the case where states are grouped according to a pre-defined clustering assignment. In this case, it is common to aggregate learned parameters according to this assignment [Poupart et al., 2006, Sorg et al., 2010, Png and Pineau, 2011]. The advantage is that there are fewer parameters to estimate, and thus, learning can be achieved with fewer samples. The main downside is that this requires a hand-coded assignment. In practice, it may also be preferable to use a coarser clustering than is strictly correct in order to improve variance (at the expense of more bias). This is a standard model selection problem.

More sophisticated approaches have been proposed to automate the process of clustering. In cases where the (unknown) model parameters

can be assumed to be sparse, it is possible to incorporate a sparseness assumption in the Dirichlet estimation through use of a hierarchical prior [Friedman and Singer, 1999]. An alternative is to maintain a posterior over the state clustering. Non-parametric models of state clustering have been considered using a Chinese Restaurant Process [Asmuth et al., 2009, Asmuth and Littman, 2011]. These could be extended to many of the other model-based BRL methods described above. A related approach was proposed for the case of partially observable MDPs, where the posterior is expressed over the set of latent (unobserved) states and is represented using a hierarchical Dirichlet process [Doshi-Velez, 2009].

A sensibly different approach proposes to express the prior over the space of policies, rather than over the space of parametric models [Doshi-Velez et al., 2010]. The goal here is to leverage trajectories acquired from an expert planner, which can be used to define this prior over policies. It is assumed that the expert knew something about the domain when computing its policy. An interesting insight is to use the infinite POMDP model [Doshi-Velez, 2009] to specify the policy prior, by simply reversing the role of actions and observations; a preference for simpler policies can be expressed by appropriately setting the hyper-parameters of the hierarchical Dirichlet process.

A few works have considered the problem of model-based BRL in cases where the underlying MDP has specific structure. In the case of factored MDPs, Ross and Pineau [2008] show that it is possible to simultaneously maintain a posterior over the structure and the parameters of the domain. The structure in this case captures the bipartite dynamic Bayes network that describes the state-to-state transitions. The prior over structures is maintained using an MCMC algorithm with appropriate graph to graph transition probabilities. The prior over model parameters is conditioned on the graph structure. Empirical results show that in some domains, it is more efficient to simultaneously estimate the structure and the parameters, rather than estimate only the parameters given a known structure.

When the model is parametrized, Gopalan and Mannor [2015] use an information theoretic approach to quickly find the set of "probable"

parameters in a pseudo-Bayesian setting. They show that the regret can be exponentially lower than models where the model is flat. Furthermore, the analysis can be done in a frequentist fashion leading eventually to a logarithmic regret.

In multi-task reinforcement learning, the goal is learn a good policy over a distribution of MDPs. A naive approach would be to assume that all observations are coming from a single model, and apply Bayesian RL to estimate this mean model. However by considering a hierarchical infinite mixture model over MDPs (represented by a hierarchical Dirichlet process), the agent is able to learn a distribution over different classes of MDPs, including estimating the number of classes and the parameters of each class [Wilson et al., 2007].

5

Model-free Bayesian Reinforcement Learning

As discussed in §2.4, model-free RL methods are those that do not explicitly learn a model of the system and only use sample trajectories obtained by direct interaction with the system. In this section, we present a family of value function Bayesian RL (BRL) methods, called Gaussian process temporal difference (GPTD) learning, and two families of policy search BRL techniques: a class of Bayesian policy gradient (BPG) algorithms and a class of Bayesian actor-critic (BAC) algorithms.

5.1 Value Function Algorithms

As mentioned in §2.4, value function RL methods search in the space of value functions, functions from the state (state-action) space to real numbers, to find the optimal value (action-value) function, and then use it to extract an optimal policy. In this section, we focus on the policy iteration (PI) approach and start by tackling the policy evaluation problem, i.e., the process of estimating the value (action-value) function V^μ (Q^μ) of a given policy μ (see §2.4). In policy evaluation, the quantity of interest is the value function of a given policy, which is unfortunately

hidden. However, a closely related random variable, the reward signal, is observable. Moreover, these hidden and observable quantities are related through the Bellman equation (Eq. 2.4). Thus, it is possible to extract information about the value function from the noisy samples of the reward signals. A Bayesian approach to this problem employs the Bayesian methodology to infer a posterior distribution over value functions, conditioned on the state-reward trajectory observed while running a MDP. Apart from the value estimates given by the posterior mean, a Bayesian solution also provides the variance of values around this mean, supplying the practitioner with an accuracy measure of the value estimates.

In the rest of this section, we study a Bayesian framework that uses a Gaussian process (GP) approach to this problem, called Gaussian process temporal difference (GPTD) learning [Engel et al., 2003, 2005a, Engel, 2005]. We then show how this Bayesian policy evaluation method (GPTD) can be used for control (to improve the policy and to eventually find a good or an optimal policy) and present a Bayesian value function RL method, called GPSARSA.

5.1.1 Gaussian Process Temporal Difference Learning

Gaussian process temporal difference (GPTD) learning [Engel et al., 2003, 2005a, Engel, 2005] is a GP-based framework that uses linear statistical models (see §2.5.2) to relate, in a probabilistic way, the underlying hidden value function with the observed rewards, for a MDP controlled by some fixed policy μ. The GPTD framework may be used with both parametric and non-parametric representations of the value function, and applied to general MDPs with infinite state and action spaces. Various flavors of the basic model yields several different online algorithms, including those designed for learning action-values, providing the basis for model-free policy improvement, and thus, full PI algorithms.

Since the focus of this section is on policy evaluation, to simplify the notation, we remove the dependency to the policy μ and use D, P, R, and V instead of D^μ, P^μ, R^μ, and V^μ. As shown in §2.2, the value V is the result of taking the expectation of the discounted return D

with respect to the randomness in the trajectory and in the rewards collected therein (Eq. 2.3). In the classic frequentist approach, V is not random, since it is the true, albeit unknown, value function of policy μ. On the other hand, in the Bayesian approach, V is viewed as a random entity by assigning it additional randomness due to our subjective uncertainty regarding the MDP's transition kernel P and reward function q (*intrinsic* uncertainty). We do not know what the true functions P and q are, which means that we are also uncertain about the true value function. We model this additional *extrinsic* uncertainty by defining V as a random process indexed by the state variable x. In order to separate the two sources of uncertainty inherent in the discounted return process D, we decompose it as

$$D(s) = \mathbf{E}[D(s)] + D(s) - \mathbf{E}[D(s)] = V(s) + \Delta V(s), \tag{5.1}$$

where

$$\Delta V(s) \stackrel{\text{def}}{=} D(s) - V(s) \tag{5.2}$$

is a zero-mean residual. When the MDP's model is known, V becomes deterministic and the randomness in D is fully attributed to the intrinsic randomness in the state-reward trajectory, modelled by ΔV. On the other hand, in a MDP in which both the transitions and rewards are deterministic but unknown, ΔV becomes zero (deterministic), and the randomness in D is due solely to the extrinsic uncertainty, modelled by V.

We write the following Bellman-like equation for D using its definition (Eq. 2.2)

$$D(s) = R(s) + \gamma D(s'), \quad s' \sim P(\cdot|s). \tag{5.3}$$

Substituting Eq. 5.1 into Eq. 5.3 and rearranging we obtain[1]

$$R(s) = V(s) - \gamma V(s') + N(s, s'), \quad s' \sim P(\cdot|s), \quad \text{where}$$
$$N(s, s') \stackrel{\text{def}}{=} \Delta V(s) - \gamma \Delta V(s'). \tag{5.4}$$

When we are provided with a system trajectory ξ of size T, we write the model-equation (5.4) for ξ, resulting in the following set of T equations

$$R(s_t) = V(s_t) - \gamma V(s_{t+1}) + N(s_t, s_{t+1}), \quad t = 0, \ldots, T-1. \tag{5.5}$$

[1]Note that in Eq. 5.4, we removed the dependency of N to policy μ and replaced N^μ with N.

By defining $\boldsymbol{R}_T = \big(R(s_0), \ldots, R(s_{T-1})\big)^\top$, $\boldsymbol{V}_{T+1} = \big(V(s_0), \ldots, V(s_T)\big)^\top$, and $\boldsymbol{N}_T = \big(N(s_0, s_1), \ldots, N(s_{T-1}, s_T)\big)^\top$, we write the above set of T equations as

$$\boldsymbol{R}_T = \boldsymbol{H}\boldsymbol{V}_{T+1} + \boldsymbol{N}_T \,, \tag{5.6}$$

where $\boldsymbol{H}$ is the following $T \times (T+1)$ matrix:

$$\boldsymbol{H} = \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 & 0 \\ 0 & 1 & -\gamma & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & -\gamma \end{bmatrix}. \tag{5.7}$$

Note that in episodic problems, if a goal-state is reached at time-step T, the final equation in (5.5) is

$$R(s_{T-1}) = V(s_{T-1}) + N(s_{T-1}, s_T), \tag{5.8}$$

and thus, Eq. 5.6 becomes

$$\boldsymbol{R}_T = \boldsymbol{H}\boldsymbol{V}_T + \boldsymbol{N}_T \,, \tag{5.9}$$

where $\boldsymbol{H}$ is the following $T \times T$ square invertible matrix with determinant equal to 1,

$$\boldsymbol{H} = \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 & 0 \\ 0 & 1 & -\gamma & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & -\gamma \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix} \text{ and } \boldsymbol{H}^{-1} = \begin{bmatrix} 1 & \gamma & \gamma^2 & \ldots & \gamma^{T-1} \\ 0 & 1 & \gamma & \ldots & \gamma^{T-2} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix}. \tag{5.10}$$

Figure 5.1 illustrates the conditional dependency relations between the latent value variables $V(s_t)$, the noise variables $\Delta V(s_t)$, and the observable rewards $R(s_t)$. Unlike the GP regression diagram of Figure 2.1, there are vertices connecting variables from different time steps, making the ordering of samples important. Since the diagram in Figure 5.1 is for the episodic setting, also note that for the last state in each episode (s_{T-1} in this figure), $R(s_{T-1})$ depends only on $V(s_{T-1})$ and $\Delta V(s_{T-1})$ (as in Eqs. 5.8 and 5.9).

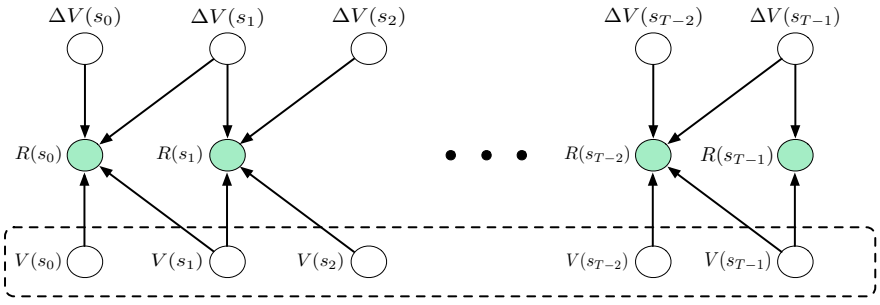

Figure 5.1: A graph illustrating the conditional independencies between the latent $V(s_t)$ value variables (bottom row), the noise variables $\Delta V(s_t)$ (top row), and the observable $R(s_t)$ reward variables (middle row), in the GPTD model. As in the case of GP regression, all of the $V(s_t)$ variables should be connected by arrows, due to the dependencies introduced by the prior. To avoid cluttering the diagram, this was marked by the dashed frame surrounding them.

In order to fully define a probabilistic generative model, we also need to specify the distribution of the noise process $\boldsymbol{N}_T$. In order to derive the noise distribution, we make the following two assumptions (see Appendix B for a discussion about these assumptions):

Assumption A2 The residuals $\Delta \boldsymbol{V}_{T+1} = \big(\Delta V(s_0), \ldots, \Delta V(s_T)\big)^\top$ can be modeled as a Gaussian process.

Assumption A3 Each of the residuals $\Delta V(s_t)$ is generated independently of all the others, i.e., $\mathbf{E}\big[\Delta V(s_i)\Delta V(s_j)\big] = 0$, for $i \neq j$.

By definition (Eq. 5.2), $\mathbf{E}\big[\Delta V(s)\big] = 0$ for all s. Using Assumption A3, it is easy to show that $\mathbf{E}\big[\Delta V(x_t)^2\big] = \mathbf{Var}\big[D(x_t)\big]$. Thus, denoting $\mathbf{Var}\big[D(x_t)\big] = \sigma_t^2$, Assumption A2 may be written as $\Delta \boldsymbol{V}_{T+1} \sim \mathcal{N}\big(\mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}_{T+1})\big)$. Since $\boldsymbol{N}_T = \boldsymbol{H}\Delta \boldsymbol{V}_{T+1}$, we have $\boldsymbol{N}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with

$$\boldsymbol{\Sigma} = \boldsymbol{H}\,\mathrm{diag}(\boldsymbol{\sigma}_{T+1})\boldsymbol{H}^\top \tag{5.11}$$
$$= \begin{bmatrix} \sigma_0^2+\gamma^2\sigma_1^2 & -\gamma\sigma_1^2 & 0 & \dots & 0 & 0 \\ -\gamma\sigma_1^2 & \sigma_1^2+\gamma^2\sigma_2^2 & -\gamma\sigma_2^2 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_{T-2}^2+\gamma^2\sigma_{T-1}^2 & -\gamma\sigma_{T-1}^2 \\ 0 & 0 & 0 & \dots & -\gamma\sigma_{T-1}^2 & \sigma_{T-1}^2+\gamma^2\sigma_T^2 \end{bmatrix}.$$

Eq. 5.11 indicates that the Gaussian noise process $\boldsymbol{N}_T$ is colored with a tri-diagonal covariance matrix. If we assume that for all $t = 0, \dots, T$, $\sigma_t = \sigma$, then $\mathrm{diag}(\boldsymbol{\sigma}_{T+1}) = \sigma^2\boldsymbol{I}$ and Eq. 5.11 may be simplified and written as

$$\boldsymbol{\Sigma} = \sigma^2\boldsymbol{H}\boldsymbol{H}^\top = \sigma^2 \begin{bmatrix} 1+\gamma^2 & -\gamma & 0 & \dots & 0 & 0 \\ -\gamma & 1+\gamma^2 & -\gamma & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1+\gamma^2 & -\gamma \\ 0 & 0 & 0 & \dots & -\gamma & 1+\gamma^2 \end{bmatrix}.$$

Eq. 5.6 (or in case of episodic problems Eq. 5.9), along with the measurement noise distribution of Eq. 5.11, and a prior distribution for V (defined either parametrically or non-parametrically, see §2.5.2), completely specify a statistical generative model relating the value and reward random processes. In order to infer value estimates from a sequence of observed rewards, Bayes' rule can be applied to this generative model to derive a posterior distribution over the value function conditioned on the observed rewards.

In the case in which we place a Gaussian prior over $\boldsymbol{V}_T$, both $\boldsymbol{V}_T$ and $\boldsymbol{N}_T$ are normally distributed, and thus, the generative model of Eq. 5.6 (Eq. 5.9) will belong to the family of linear statistical models discussed in §2.5.2. Consequently, both parametric and non-parametric treatments of this model (see §2.5.2) may be applied in full to the generative model of Eq. 5.6 (Eq. 5.9), with $\boldsymbol{H}$ given by Eq. 5.7 (Eq. 5.10). Figure 5.2 demonstrates how the GPTD model described in this section is related to the family of linear statistical models and GP regression discussed in §2.5.2.

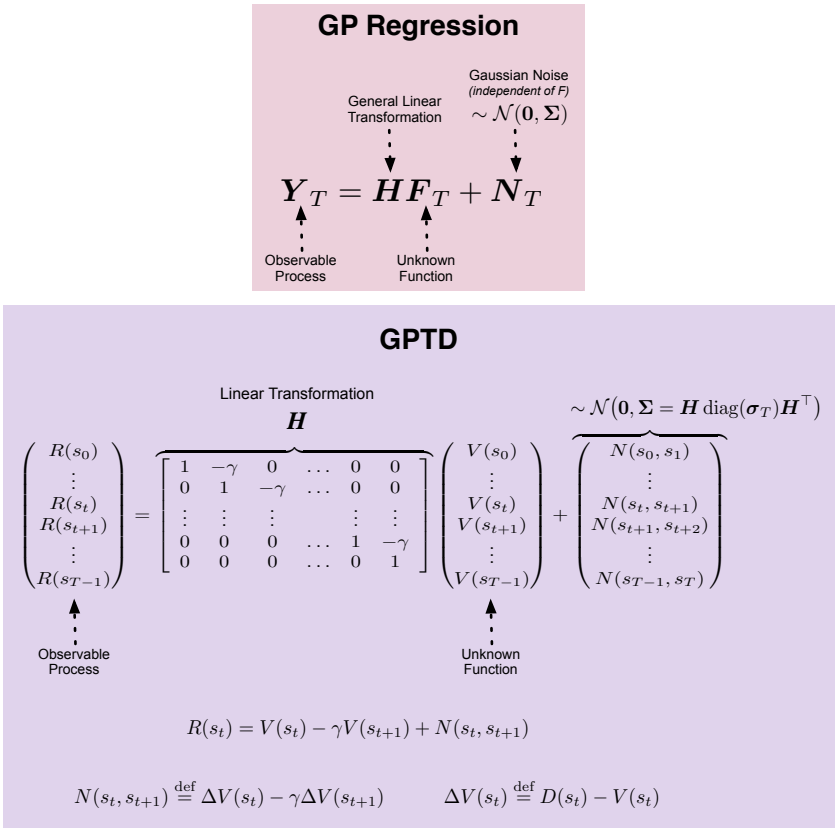

Figure 5.2: The connection between the GPTD model described in §5.1.1 and the family of linear statistical models and GP regression discussed in §2.5.2.

We are now in a position to write a closed form expression for the posterior moments of V, conditioned on an observed sequence of rewards $\boldsymbol{r}_T = \big(r(s_0), \dots, r(s_{T-1})\big)^\top$, and derive GPTD-based algorithms for value function estimation. We refer to this family of algorithms as Monte-Carlo GPTD (MC-GPTD) algorithms.

Parametric Monte-Carlo GPTD Learning: In the parametric setting, the value process is parameterized as $V(s) = \boldsymbol{\phi}(s)^\top \boldsymbol{W}$, and therefore, $\boldsymbol{V}_{T+1} = \boldsymbol{\Phi}^\top \boldsymbol{W}$ with $\boldsymbol{\Phi} = [\boldsymbol{\phi}(s_0), \dots, \boldsymbol{\phi}(s_T)]$. As in §2.5.2, if

we use the prior distribution $\boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, the posterior moments of $\boldsymbol{W}$ are given by

$$\begin{aligned} \mathbf{E}[\boldsymbol{W}|\boldsymbol{R}_T = \boldsymbol{r}_T] &= \Delta\boldsymbol{\Phi}(\Delta\boldsymbol{\Phi}^\top\Delta\boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1}\boldsymbol{r}_T, \\ \mathbf{Cov}[\boldsymbol{W}|\boldsymbol{R}_T = \boldsymbol{r}_T] &= \boldsymbol{I} - \Delta\boldsymbol{\Phi}(\Delta\boldsymbol{\Phi}^\top\Delta\boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1}\Delta\boldsymbol{\Phi}^\top, \end{aligned} \tag{5.12}$$

where $\Delta\boldsymbol{\Phi} = \boldsymbol{\Phi}\boldsymbol{H}^\top$. To have a smaller matrix inversion, Eq. 5.12 may be written as

$$\begin{aligned} \mathbf{E}[\boldsymbol{W}|\boldsymbol{R}_T = \boldsymbol{r}_T] &= (\Delta\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\Delta\boldsymbol{\Phi}^\top + \boldsymbol{I})^{-1}\Delta\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{r}_T, \\ \mathbf{Cov}[\boldsymbol{W}|\boldsymbol{R}_T = \boldsymbol{r}_T] &= (\Delta\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\Delta\boldsymbol{\Phi}^\top + \boldsymbol{I})^{-1}. \end{aligned} \tag{5.13}$$

Note that in episodic problems $\boldsymbol{H}$ is invertible, and thus, assuming a constant noise variance, i.e., $\mathrm{diag}(\boldsymbol{\sigma}_T) = \sigma^2\boldsymbol{I}$, Eq. 5.13 becomes

$$\begin{aligned} \mathbf{E}[\boldsymbol{W}|\boldsymbol{R}_T = \boldsymbol{r}_T] &= (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{\Phi}\boldsymbol{H}^{-1}\boldsymbol{r}_T, \\ \mathbf{Cov}[\boldsymbol{W}|\boldsymbol{R}_T = \boldsymbol{r}_T] &= \sigma^2(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma^2\boldsymbol{I})^{-1}. \end{aligned} \tag{5.14}$$

Eq. 5.14 is equivalent to Eq. 2.22 with $\boldsymbol{y}_T = \boldsymbol{H}^{-1}\boldsymbol{r}_T$ (see the discussion of Assumption A3).

Using Eq. 5.13 (or Eq. 5.14), it is possible to derive both a batch algorithm [Engel, 2005, Algorithm 17] and a recursive online algorithm [Engel, 2005, Algorithm 18] to compute the posterior moments of the weight vector $\boldsymbol{W}$, and thus, the value function $V(\cdot) = \boldsymbol{\phi}(\cdot)^\top\boldsymbol{W}$.

Non-parametric Monte-Carlo GPTD Learning: In the non-parametric case, we bypass the parameterization of the value process by placing a prior directly over the space of value functions. From §2.5.2 with the GP prior $\boldsymbol{V}_{T+1} \in \mathcal{N}(\mathbf{0}, \boldsymbol{K})$, we obtain

$$\begin{aligned} \mathbf{E}[V(s)|\boldsymbol{R}_T = \boldsymbol{r}_T] &= \boldsymbol{k}(s)^\top\boldsymbol{\alpha}, \\ \mathbf{Cov}[V(s), V(s')|\boldsymbol{R}_T = \boldsymbol{r}_T] &= k(s,s') - \boldsymbol{k}(s)^\top\boldsymbol{C}\boldsymbol{k}(s'), \end{aligned} \tag{5.15}$$

where

$$\boldsymbol{\alpha} = \boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1}\boldsymbol{r}_T \quad \text{and} \quad \boldsymbol{C} = \boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma})^{-1}\boldsymbol{H}. \tag{5.16}$$

Similar to the parametric case, assuming a constant noise variance, Eq. 5.16 may be written for episodic problems as

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{H}^{-1} \boldsymbol{r}_T \qquad \text{and} \qquad \boldsymbol{C} = (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1}. \quad (5.17)$$

Eq. 5.17 is equivalent to Eq. 2.18 with $\boldsymbol{y}_T = \boldsymbol{H}^{-1} \boldsymbol{r}_T$ (see the discussion of Assumption A3).

As in the parametric case, it is possible to derive recursive updates for $\boldsymbol{\alpha}$ and $\boldsymbol{C}$, and an online algorithm [Engel, 2005, Algorithm 19] to compute the posterior moments of the value function V.

Sparse Non-parametric Monte-Carlo GPTD Learning: In the parametric case, the computation of the posterior may be performed online in $O(n^2)$ time per sample and $O(n^2)$ memory, where n is the number of basis functions used to approximate V. In the non-parametric case, we have a new basis function for each new sample we observe, making the cost of adding the t'th sample $O(t^2)$ in both time and memory. This would seem to make the non-parametric form of GPTD computationally infeasible except in small and simple problems. However, the computational cost of non-parametric GPTD can be reduced by using an online sparsification method ([Engel et al., 2002], [Engel, 2005, Chapter 2]), to a level where it can be efficiently implemented online. In many cases, this results in significant computational savings, both in terms of memory and time, when compared to the non-sparse solution. For the resulting algorithm, we refer the readers to the sparse non-parametric GPTD algorithm in [Engel et al., 2005a, Table 1] or [Engel, 2005, Algorithm 20].

5.1.2 Connections with Other TD Methods

In §5.1.1, we showed that the stochastic GPTD model is equivalent to GP regression on MC samples of the discounted return (see the discussion of Assumption A3). Engel [Engel, 2005] showed that by suitably selecting the noise covariance matrix $\boldsymbol{\Sigma}$ in the stochastic GPTD model, it is possible to obtain GP-based variants of LSTD(λ) algorithm [Bradtke

and Barto, 1996, Boyan, 1999]. The main idea is to obtain the value of the weight vector $\boldsymbol{W}$ in the parametric GPTD model by carrying out maximum likelihood (ML) inference ($\boldsymbol{W}$ is the value for which the observed data is most likely to be generated by the stochastic GPTD model). This allows us to derive LSTD(λ) for each value of $\lambda \leq 1$, as an ML solution arising from some GPTD generative model with a specific noise covariance matrix $\boldsymbol{\Sigma}$.

5.1.3 Policy Improvement with GPSARSA

SARSA [Rummery and Niranjan, 1994] is a straightforward extension of the TD algorithm [Sutton, 1988] to control, in which action-values are estimated. This allows policy improvement steps to be performed without requiring any additional knowledge of the MDP model. The idea is that under the policy μ, we may define a process with state space $\mathcal{Z} = \mathcal{S} \times \mathcal{A}$ (the space of state-action pairs) and the reward model of the MDP. This process is Markovian with transition density $P^{\mu}(z'|z) = P^{\mu}(s'|s)\mu(a'|s')$, where $z = (s, a)$ and $z' = (s', a')$ (see §2.2 for more details on this process). SARSA is simply the TD algorithm applied to this process. The same reasoning may be applied to derive a GPSARSA algorithm from a GPTD algorithm. This is equivalent to rewriting the model-equations of §5.1.1 for the action-value function Q^{μ}, which is a function of z, instead of the value function V^{μ}, a function of s.

In the non-parametric setting, we simple need to define a covariance kernel function over state-action pairs, i.e., $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$. Since states and actions are different entities, it makes sense to decompose k into a state-kernel k_s and an action-kernel k_a: $k(z, z') = k_s(s, s')k_a(a, a')$. If both k_s and k_a are kernels, we know that k is also a kernel [Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004]. The state and action kernels encode our prior beliefs on value correlations between different states and actions, respectively. All that remains is to run the GPTD algorithms (sparse, batch, online) on the state-action-reward sequence using the new state-action kernel. Action selection may be performed by the ϵ-greedy exploration strategy. The main difficulty that may arise here is in finding the greedy action from a large or

infinite number of possible actions (when $|\mathcal{A}|$ is large or infinite). This may be solved by sampling the value estimates for a few randomly chosen actions and find the greedy action among only them. However, an ideal approach would be to design the action-kernel in such a way as to provide a closed-form expression for the greedy action. Similar to the non-parametric setting, in the parametric case, deriving GPSARSA algorithms from their GPTD counterparts is still straightforward; we just need to define a set of basis functions over $\mathcal{Z}$ and use them to approximate action-value functions.

5.1.4 Related Work

Another approach to employ GPs in RL was proposed by Rasmussen and Kuss [Rasmussen and Kuss, 2004]. The approach taken in this work is notably different from the generative approach of the GPTD framework. Two GPs are used in [Rasmussen and Kuss, 2004], one to learn the MDP's transition model and one to estimate the value function. This leads to an inherently offline algorithm. There are several other limitations to this framework. First, the state dynamics is assumed to be factored, in the sense that each state coordinate evolves in time independently of all others. This is a rather strong assumption that is not likely to be satisfied in most real problems. Second, it is assumed that the reward function is completely known in advance, and is of a very special form – either polynomial or Gaussian. Third, the covariance kernels used are also restricted to be either polynomial or Gaussian or a mixture of the two, due to the need to integrate over products of GPs. Finally, the value function is only modelled at a predefined set of support states, and is solved only for them. No method is proposed to ensure that this set of states is representative in any way.

Similar to most kernel-based methods, the choice of prior distribution significantly affects the empirical performance of the learning algorithm. Reisinger et al. [Reisinger et al., 2008] proposed an online model (prior covariance function) selection method for GPTD using sequential Monte-Carlo methods and showed that their method yields better asymptotic performance than standard GPTD for many different kernel families.

5.2 Bayesian Policy Gradient

Policy gradient (PG) methods are RL algorithms that maintain a class of smoothly parameterized stochastic policies $\{\mu(\cdot|s;\boldsymbol{\theta}), s \in \mathcal{S}, \boldsymbol{\theta} \in \Theta\}$, and update the policy parameters $\boldsymbol{\theta}$ by adjusting them in the direction of an estimate of the gradient of a performance measure (e.g., [Williams, 1992, Marbach, 1998, Baxter and Bartlett, 2001]). As an illustration of such a parameterized policy, consider the following example:

Example 5.1 (Online shop – parameterized policy)**.** Recall the online shop domain of Example 1.1. Assume that for each customer state $s \in \mathcal{X}$, and advertisement $a \in \mathcal{A}$, we are given a set of n numeric values $\phi(s,a) \in \mathbb{R}^n$ that represent some features of the state and action. For example, these features could be the number of items in the cart, the average price of items in the cart, the age of the customer, etc. A popular parametric policy representation is the softmax policy, defined as

$$\mu(a|s;\boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\theta}^\top \phi(s,a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\boldsymbol{\theta}^\top \phi(s,a')\right)}.$$

The intuition behind this policy, is that if $\boldsymbol{\theta}$ is chosen such that $\boldsymbol{\theta}^\top \phi(s,a)$ is an approximation of the state-action value function $Q(s,a)$, then the softmax policy is an approximation of the greedy policy with respect to Q – which is the optimal policy.

The performance of a policy μ is often measured by its *expected return*, $\eta(\mu)$, defined by Eq. 2.1. Since in this setting a policy μ is represented by its parameters $\boldsymbol{\theta}$, policy dependent functions such as $\eta(\mu)$ and $\Pr(\xi;\mu)$ may be written as $\eta(\boldsymbol{\theta})$ and $\Pr(\xi;\boldsymbol{\theta})$.

The *score function* or *likelihood ratio* method has become the most prominent technique for gradient estimation from simulation (e.g., Glynn [1990], Williams [1992]). This method estimates the gradient of the expected return with respect to the policy parameters $\boldsymbol{\theta}$, defined by Eq. 2.1, using the following equation:[2]

$$\nabla \eta(\boldsymbol{\theta}) = \int \bar{\rho}(\xi) \frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})} \Pr(\xi;\boldsymbol{\theta}) d\xi. \tag{5.18}$$

[2]We use the notation ∇ to denote $\nabla_{\boldsymbol{\theta}}$ – the gradient with respect to the policy parameters.

In Eq. 5.18, the quantity

$$\begin{aligned} \boldsymbol{u}(\xi;\boldsymbol{\theta}) &= \frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})} = \nabla \log \Pr(\xi;\boldsymbol{\theta}) \\ &= \sum_{t=0}^{T-1} \frac{\nabla \mu(a_t|s_t;\boldsymbol{\theta})}{\mu(a_t|s_t;\boldsymbol{\theta})} = \sum_{t=0}^{T-1} \nabla \log \mu(a_t|s_t;\boldsymbol{\theta}) \end{aligned} \tag{5.19}$$

is called the (Fisher) score function or likelihood ratio of trajectory ξ under policy $\boldsymbol{\theta}$. Most of the work on PG has used classical Monte-Carlo (MC) to estimate the (integral) gradient in Eq. 5.18. These methods (in their simplest form) generate M i.i.d. sample paths $\xi_1, \ldots, \xi_M$ according to $\Pr(\xi;\boldsymbol{\theta})$, and estimate the gradient $\nabla\eta(\boldsymbol{\theta})$ using the *unbiased* MC estimator

$$\widehat{\nabla\eta}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \rho(\xi_i) \nabla \log \Pr(\xi_i;\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \rho(\xi_i) \sum_{t=0}^{T_i-1} \nabla \log \mu(a_{t,i}|s_{t,i};\boldsymbol{\theta}). \tag{5.20}$$

Both theoretical results and empirical evaluations have highlighted a major shortcoming of these algorithms: namely, the high variance of the gradient estimates, which in turn results in sample-inefficiency (e.g., Marbach [1998], Baxter and Bartlett [2001]). Many solutions have been proposed for this problem, each with its own pros and cons. In this section, we describe Bayesian policy gradient (BPG) algorithms that tackle this problem by using a Bayesian alternative to the MC estimation of Eq. 5.20 [Ghavamzadeh and Engel, 2006]. These BPG algorithms are based on *Bayesian quadrature*, i.e., a Bayesian approach to integral evaluation, proposed by O'Hagan [O'Hagan, 1991].[3] The algorithms use Gaussian processes (GPs) to define a prior distribution over the gradient of the expected return, and compute its posterior conditioned on the observed data. This reduces the number of samples needed to obtain accurate (integral) gradient estimates. Moreover, estimates of the natural gradient as well as a measure of the uncertainty in the gradient estimates, namely the gradient covariance, are provided at little extra cost. In the next two sections, we first briefly describe

[3] O'Hagan [O'Hagan, 1991] mentions that this approach may be traced even as far back as the work by Poincaré [Poincaré, 1896].

Bayesian quadrature and then present the BPG models and algorithms.

5.2.1 Bayesian Quadrature

Bayesian quadrature (BQ) [O'Hagan, 1991], as its name suggests, is a Bayesian method for evaluating an integral using samples of its integrand. Let us consider the problem of evaluating the following integral[4]

$$\zeta = \int f(x)g(x)dx. \tag{5.21}$$

BQ is based on the following reasoning: In the Bayesian approach, $f(\cdot)$ is random simply because it is unknown. We are therefore uncertain about the value of $f(x)$ until we actually evaluate it. In fact, even then, our uncertainty is not always completely removed, since measured samples of $f(x)$ may be corrupted by noise. Modeling f as a GP means that our uncertainty is completely accounted for by specifying a Normal prior distribution over the functions, $f(\cdot) \sim \mathcal{N}\big(\bar{f}(\cdot), k(\cdot,\cdot)\big)$, i.e.,

$$\mathbf{E}\big[f(x)\big] = \bar{f}(x) \quad \text{and} \quad \mathbf{Cov}\big[f(x), f(x')\big] = k(x,x'), \quad \forall x, x' \in \mathcal{X}.$$

The choice of kernel function k allows us to incorporate prior knowledge on the smoothness properties of the integrand into the estimation procedure. When we are provided with a set of samples $\mathcal{D}_M = \big\{(x_i, y(x_i))\big\}_{i=1}^{M}$, where $y(x_i)$ is a (possibly noisy) sample of $f(x_i)$, we apply Bayes' rule to condition the prior on these sampled values. If the measurement noise is normally distributed, the result is a Normal posterior distribution of $f|\mathcal{D}_M$. The expressions for the posterior mean and covariance are standard (see Eqs. 2.14–2.16):

$$\begin{aligned} \mathbf{E}\big[f(x)|\mathcal{D}_M\big] &= \bar{f}(x) + \boldsymbol{k}(x)^\top \boldsymbol{C}(\boldsymbol{y} - \bar{\boldsymbol{f}}), \\ \mathbf{Cov}\big[f(x), f(x')|\mathcal{D}_M\big] &= k(x,x') - \boldsymbol{k}(x)^\top \boldsymbol{C}\boldsymbol{k}(x'), \end{aligned} \tag{5.22}$$

where $\boldsymbol{K}$ is the kernel (or Gram) matrix, and $[\boldsymbol{\Sigma}]_{i,j}$ is the measurement noise covariance between the ith and jth samples. It is typically

[4]Similar to [O'Hagan, 1991], here for simplicity we consider the case where the integral to be estimated is a scalar-valued integral. However, the results of this section can be extended to vector-valued integrals, such as the gradient of the expected return with respect to the policy parameters that we shall study in §5.2.2 (see Ghavamzadeh et al. [2013]).

assumed that the measurement noise is i.i.d., in which case $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$, where σ^2 is the noise variance and $\boldsymbol{I}$ is the $(M \times M)$ identity matrix.

$$\bar{\boldsymbol{f}} = \big(\bar{f}(x_1), \ldots, \bar{f}(x_M)\big)^\top, \quad \boldsymbol{k}(x) = \big(k(x_1, x), \ldots, k(x_M, x)\big)^\top,$$
$$\boldsymbol{y} = \big(y(x_1), \ldots, y(x_M)\big)^\top, \quad [\boldsymbol{K}]_{i,j} = k(x_i, x_j), \quad \boldsymbol{C} = (\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}.$$

Since integration is a linear operation, the posterior distribution of the integral in Eq. 5.21 is also Gaussian, and the posterior moments are given by [O'Hagan, 1991]

$$\begin{aligned} \mathbf{E}[\zeta|\mathcal{D}_M] &= \int \mathbf{E}[f(x)|\mathcal{D}_M] g(x) dx, \\ \mathbf{Var}[\zeta|\mathcal{D}_M] &= \iint \mathbf{Cov}[f(x), f(x')|\mathcal{D}_M] g(x) g(x') dx dx'. \end{aligned} \tag{5.23}$$

Substituting Eq. 5.22 into Eq. 5.23, we obtain

$$\mathbf{E}[\zeta|\mathcal{D}_M] = \zeta_0 + \boldsymbol{b}^\top \boldsymbol{C}(\boldsymbol{y} - \bar{\boldsymbol{f}}) \qquad \text{and} \qquad \mathbf{Var}[\zeta|\mathcal{D}_M] = b_0 - \boldsymbol{b}^\top \boldsymbol{C} \boldsymbol{b},$$

where we made use of the definitions

$$\zeta_0 = \int \bar{f}(x) g(x) dx \ , \ \boldsymbol{b} = \int \boldsymbol{k}(x) g(x) dx \ , \ b_0 = \iint k(x, x') g(x) g(x') dx dx'. \tag{5.24}$$

Note that ζ_0 and b_0 are the prior mean and variance of ζ, respectively. It is important to note that in order to prevent the problem from "degenerating into infinite regress", as phrased by O'Hagan [O'Hagan, 1991], we should decompose the integrand into parts: f (the GP part) and g, and select the GP prior, i.e., prior mean $\bar{f}$ and covariance k, so as to allow us to solve the integrals in Eq. 5.24 analytically. Otherwise, we begin with evaluating one integral (Eq. 5.21) and end up with evaluating three integrals (Eq. 5.24).

5.2.2 Bayesian Policy Gradient Algorithms

In this section, we describe the Bayesian policy gradient (BPG) algorithms [Ghavamzadeh and Engel, 2006]. These algorithms use Bayesian quadrature (BQ) to estimate the gradient of the expected return with respect to the policy parameters (Eq. 5.18). In the Bayesian approach, the expected return of the policy characterized by the parameters $\boldsymbol{\theta}$

$$\eta_B(\boldsymbol{\theta}) = \int \bar{\rho}(\xi) \Pr(\xi; \boldsymbol{\theta}) d\xi \tag{5.25}$$

is a random variable because of our subjective Bayesian uncertainty concerning the process generating the return. Under the quadratic loss, the optimal Bayesian performance measure is the posterior expected value of $\eta_B(\boldsymbol{\theta})$, $\mathbf{E}[\eta_B(\boldsymbol{\theta})|\mathcal{D}_M]$. However, since we are interested in optimizing the performance rather than in evaluating it, we would rather evaluate the posterior mean of the gradient of $\eta_B(\boldsymbol{\theta})$ with respect to the policy parameters $\boldsymbol{\theta}$, i.e.,

$$\begin{aligned}\nabla\mathbf{E}[\eta_B(\boldsymbol{\theta})|\mathcal{D}_M] &= \mathbf{E}[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M] \\ &= \mathbf{E}\left[\int \bar{\rho}(\xi)\frac{\nabla \Pr(\xi;\boldsymbol{\theta})}{\Pr(\xi;\boldsymbol{\theta})}\Pr(\xi;\boldsymbol{\theta})d\xi\Big|\mathcal{D}_M\right].\end{aligned} \tag{5.26}$$

Gradient Estimation: In BPG, we cast the problem of estimating the gradient of the expected return (Eq. 5.26) in the form of Eq. 5.21 and use the BQ approach described in §5.2.1. We partition the integrand into two parts, $f(\xi;\boldsymbol{\theta})$ and $g(\xi;\boldsymbol{\theta})$, model f as a GP, and assume that g is a function known to us. We then proceed by calculating the posterior moments of the gradient $\nabla\eta_B(\boldsymbol{\theta})$ conditioned on the observed data. Ghavamzadeh and Engel [Ghavamzadeh and Engel, 2006] proposed two different ways of partitioning the integrand in Eq. 5.26, resulting in the two distinct Bayesian models summarized in Table 5.1 (see Ghavamzadeh et al. [2013], for more details). Figure 5.3 shows how the BQ approach has been used in each of the two BPG models of [Ghavamzadeh and Engel, 2006], summarized in Table 5.1, as well as in the Bayesian actor-critic (BAC) formulation of §5.3.

It is important to note that in both models $\bar{\rho}(\xi)$ belongs to the GP part, i.e., $f(\xi;\boldsymbol{\theta})$. This is because in general, $\bar{\rho}(\xi)$ cannot be known exactly, even for a given ξ (due to the stochasticity of the rewards), but can be estimated for a sample trajectory ξ. The more important and rather critical point is that $\Pr(\xi;\boldsymbol{\theta})$ cannot belong to either the $f(\xi;\boldsymbol{\theta})$ or $g(\xi;\boldsymbol{\theta})$ part of the model, since it is not known (in model-free setting) and cannot be estimated for a sample trajectory ξ. However, Ghavamzadeh and Engel [Ghavamzadeh and Engel, 2006] showed that interestingly, it is sufficient to assign $\Pr(\xi;\boldsymbol{\theta})$ to the $g(\xi;\boldsymbol{\theta})$ part of the model and use an appropriate Fisher kernel (see Table 5.1 for the kernels), rather than having exact knowledge of $\Pr(\xi;\boldsymbol{\theta})$

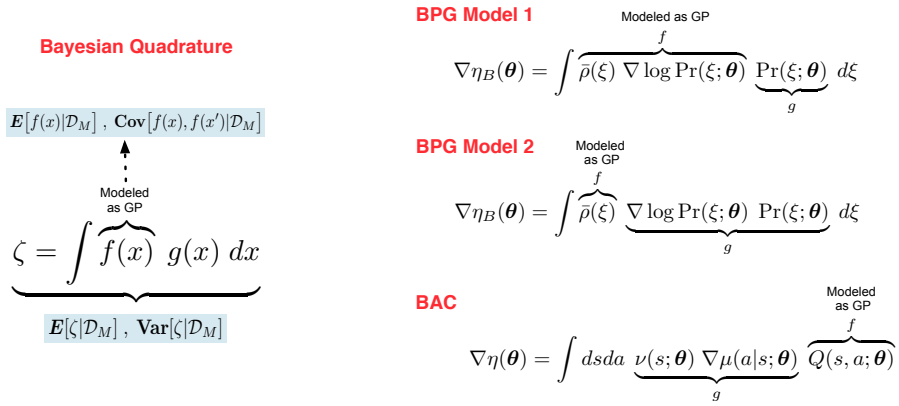

Figure 5.3: The connection between BQ and the two BPG models of [Ghavamzadeh and Engel, 2006], and the Bayesian actor-critic (BAC) formulation of §5.3.

	Model 1	**Model 2**
Deter. factor (g)	$g(\xi;\boldsymbol{\theta}) = \Pr(\xi;\boldsymbol{\theta})$	$g(\xi;\boldsymbol{\theta}) = \nabla \Pr(\xi;\boldsymbol{\theta})$
GP factor (f)	$f(\xi;\boldsymbol{\theta}) = \bar{\rho}(\xi)\nabla \log \Pr(\xi;\boldsymbol{\theta})$	$f(\xi) = \bar{\rho}(\xi)$
Measurement (y)	$y(\xi;\boldsymbol{\theta}) = \rho(\xi)\nabla \log \Pr(\xi;\boldsymbol{\theta})$	$y(\xi) = \rho(\xi)$
Prior mean of f	$\mathbf{E}[f_j(\xi;\boldsymbol{\theta})] = 0$	$\mathbf{E}[f(\xi)] = 0$
Prior Cov. of f	$\mathbf{Cov}[f_j(\xi;\boldsymbol{\theta}), f_\ell(\xi';\boldsymbol{\theta})] = \delta_{j,\ell}\, k(\xi,\xi')$	$\mathbf{Cov}[f(\xi), f(\xi')] = k(\xi,\xi')$
Kernel function	$k(\xi,\xi') = \left(1 + \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi')\right)^2$	$k(\xi,\xi') = \boldsymbol{u}(\xi)^\top \boldsymbol{G}^{-1}\boldsymbol{u}(\xi')$
$\mathbf{E}[\nabla\eta_B(\boldsymbol{\theta})\|\mathcal{D}_M]$	$\boldsymbol{YCb}$	$\boldsymbol{BCy}$
$\mathbf{Cov}[\nabla\eta_B(\boldsymbol{\theta})\|\mathcal{D}_M]$	$(b_0 - \boldsymbol{b}^\top\boldsymbol{Cb})\boldsymbol{I}$	$\boldsymbol{B}_0 - \boldsymbol{BCB}^\top$
$\boldsymbol{b}$ or $\boldsymbol{B}$	$(\boldsymbol{b})_i = 1 + \boldsymbol{u}(\xi_i)^\top\boldsymbol{G}^{-1}\boldsymbol{u}(\xi_i)$	$\boldsymbol{B} = \boldsymbol{U}$
b_0 or $\boldsymbol{B}_0$	$b_0 = 1 + n$	$\boldsymbol{B}_0 = \boldsymbol{G}$

Table 5.1: Summary of the Bayesian policy gradient Models 1 and 2.

itself (see Ghavamzadeh et al. [2013] for more details).

In **Model 1**, a vector-valued GP prior is placed over $f(\xi;\boldsymbol{\theta})$. This induces a GP prior over the corresponding noisy measurement $y(\xi;\boldsymbol{\theta})$. It is assumed that each component of $f(\xi;\boldsymbol{\theta})$ may be evaluated inde-

pendently of all other components, and thus, the same kernel function $\boldsymbol{K}$ and noise covariance $\boldsymbol{\Sigma}$ are used for all components of $f(\xi;\boldsymbol{\theta})$. Hence for the jth component of f and y we have a priori

$$\begin{aligned}\boldsymbol{f}_j &= \left(f_j(\xi_1;\boldsymbol{\theta}),\ldots,f_j(\xi_M;\boldsymbol{\theta})\right)^\top \sim \mathcal{N}(\mathbf{0},\boldsymbol{K}),\\ \boldsymbol{y}_j &= \left(y_j(\xi_1;\boldsymbol{\theta}),\ldots,y_j(\xi_M;\boldsymbol{\theta})\right)^\top \sim \mathcal{N}(\mathbf{0},\boldsymbol{K}+\boldsymbol{\Sigma}).\end{aligned}$$

This vector-valued GP model gives us the posterior mean and covariance of $\nabla\eta_B(\boldsymbol{\theta})$ reported in Table 5.1m in which $\boldsymbol{Y} = [\boldsymbol{y}_1^\top;\ldots;\boldsymbol{y}_n^\top]$, $\boldsymbol{C} = (\boldsymbol{K}+\boldsymbol{\Sigma})^{-1}$, n is the number of policy parameters, and $\boldsymbol{G}(\boldsymbol{\theta})$ is the Fisher information matrix of policy $\boldsymbol{\theta}$ defined as[5]

$$\boldsymbol{G}(\boldsymbol{\theta}) = \mathbf{E}\big[\boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top\big] = \int \boldsymbol{u}(\xi)\boldsymbol{u}(\xi)^\top \Pr(\xi;\boldsymbol{\theta})d\xi, \tag{5.27}$$

where $\boldsymbol{u}(\xi)$ is the score function of trajectory ξ defined by Eq. 5.19. Note that the choice of the kernel k allows us to derive closed form expressions for $\boldsymbol{b}$ and b_0, and, as a result k is the quadratic Fisher kernel for the posterior moments of the gradient (see Table 5.1).

In **Model 2**, g is a vector-valued function and f is a scalar valued GP representing the expected return of the path given as its argument. The noisy measurement corresponding to $f(\xi_i)$ is $y(\xi_i) = \rho(\xi_i)$, namely, the actual return accrued while following the path ξ_i. This GP model gives us the posterior mean and covariance of $\nabla\eta_B(\boldsymbol{\theta})$ reported in Table 5.1 in which $\boldsymbol{y} = \left(\rho(\xi_1),\ldots,\rho(\xi_M)\right)^\top$ and $\boldsymbol{U} = [\boldsymbol{u}(\xi_1),\ldots,\boldsymbol{u}(\xi_M)]$. Here the choice of kernel k allows us to derive closed-form expressions for $\boldsymbol{B}$ and $\boldsymbol{B}_0$, and as a result, k is again the Fisher kernel for the posterior moments of the gradient (see Table 5.1).

Note that the choice of Fisher-type kernels is motivated by the notion that a good representation should depend on the process generating the data (see Jaakkola and Haussler [1999], Shawe-Taylor and Cristianini [2004], for a thorough discussion). The particular selection of linear and quadratic Fisher kernels is guided by the desideratum that the posterior moments of the gradient be analytically tractable as discussed at the end of §5.2.1.

[5]To simplify notation, we omit $\boldsymbol{G}$'s dependence on the policy parameters $\boldsymbol{\theta}$, and denote $\boldsymbol{G}(\boldsymbol{\theta})$ as $\boldsymbol{G}$ in the rest of this section.

The above two BPG models can define Bayesian algorithms for evaluating the gradient of the expected return with respect to the policy parameters (see [Ghavamzadeh and Engel, 2006] and [Ghavamzadeh et al., 2013] for the pseudo-code of the resulting algorithms). It is important to note that computing the quadratic and linear Fisher kernels used in Models 1 and 2 requires calculating the Fisher information matrix $\boldsymbol{G}(\boldsymbol{\theta})$ (Eq. 5.27). Consequently, every time the policy parameters are updated, $\boldsymbol{G}$ needs to be recomputed. Ghavamzadeh and Engel [Ghavamzadeh and Engel, 2006] suggest two possible approaches for online estimation of the Fisher information matrix.

Similar to most non-parametric methods, the Bayesian gradient evaluation algorithms can be made more efficient, both in time and memory, by *sparsifying* the solution. Sparsification also helps to numerically stabilize the algorithms when the kernel matrix is singular, or nearly so. Ghavamzadeh et al. [Ghavamzadeh et al., 2013] show how one can incrementally perform such sparsification in their Bayesian gradient evaluation algorithms, i.e., how to selectively add a new observed path to a set of *dictionary* paths used as a basis for representing or approximating the full solution.

Policy Improvement: Ghavamzadeh and Engel [Ghavamzadeh and Engel, 2006] also show how their Bayesian algorithms for estimating the gradient can be used to define a Bayesian policy gradient algorithm. The algorithm starts with an initial set of policy parameters $\boldsymbol{\theta}_0$, and at each iteration j, updates the parameters in the direction of the posterior mean of the gradient of the expected return $\mathbf{E}[\nabla \eta_B(\boldsymbol{\theta}_j)|\mathcal{D}_M]$ estimated by their Bayesian gradient evaluation algorithms. This is repeated N times, or alternatively, until the gradient estimate is sufficiently close to zero. Since the Fisher information matrix, $\boldsymbol{G}$, and the posterior distribution (both mean and covariance) of the gradient of the expected return are estimated at each iteration of this algorithm, we may make the following modifications in the resulting Bayesian policy gradient algorithm at little extra cost:

- Update the policy parameters in the direction of the natural gradient, $\boldsymbol{G}(\boldsymbol{\theta})^{-1}\mathbf{E}[\nabla \eta_B(\boldsymbol{\theta})|\mathcal{D}_M]$, instead of the regular gradient,

$\mathbf{E}[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M]$.

- Use the posterior covariance of the gradient of the expected return as a measure of the uncertainty in the gradient estimate, and thus, as a measure to tune the step-size parameter in the gradient update (the larger the posterior variance the smaller the step-size) (see the experiments in Ghavamzadeh et al. [2013] for more details). In a similar approach, Vien et al. [Vien et al., 2011] used BQ to estimate the Hessian matrix distribution and then used its mean as learning rate schedule to improve the performance of BPG. They empirically showed that their method performs better than BPG and BPG with natural gradient in terms of convergence speed.

It is important to note that similar to the gradient estimated by the GPOMDP algorithm of Baxter and Bartlett [Baxter and Bartlett, 2001], the gradient estimated by these algorithms, $\mathbf{E}[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M]$, can be used with the conjugate-gradient and line-search methods of Baxter and Bartlett [Baxter et al., 2001] for improved use of gradient information. This allows us to exploit the information contained in the gradient estimate more aggressively than by simply adjusting the parameters by a small amount in the direction of $\mathbf{E}[\nabla\eta_B(\boldsymbol{\theta})|\mathcal{D}_M]$.

The experiments reported in [Ghavamzadeh et al., 2013] indicate that the BPG algorithm tends to significantly reduce the number of samples needed to obtain accurate gradient estimates. Thus, given a fixed number of samples per iteration, finds a better policy than MC-based policy gradient methods. These results are in line with previous work on BQ, for example a work by Rasmussen and Ghahramani [Rasmussen and Ghahramani, 2003] that demonstrates how BQ, when applied to the evaluation of an expectation, can outperform MC estimation by orders of magnitude in terms of the mean-squared error.

Extension to Partially Observable Markov Decision Processes: The above models and algorithms can be easily extended to partially observable problems without any changes using similar techniques as in Section 6 of [Baxter and Bartlett, 2001]. This is due to the fact that

the BPG framework considers complete system trajectories as its basic observable unit, and thus, does not require the dynamic within each trajectory to be of any special form. This generality has the downside that it cannot take advantage of the Markov property when the system is Markovian (see Ghavamzadeh et al. [2013] for more details). To address this issue, Ghavamzadeh and Engel [Ghavamzadeh and Engel, 2007] then extended their BPG framework to actor-critic algorithms and present a new Bayesian take on the actor-critic architecture, which is the subject of the next section.

5.3 Bayesian Actor-Critic

Another approach to reduce the variance of the policy gradient estimates is to use an explicit representation for the value function of the policy. This class of PG algorithms are called *actor-critic* and they were among the earliest to be investigated in RL [Barto et al., 1983, Sutton, 1984]. Unlike in §5.2 where we consider complete trajectories as the basic observable unit, in this section we assume that the system is Markovian, and thus, the basic observable unit is one step system transition (state, action, next state). It can be shown that under certain regularity conditions [Sutton et al., 2000], the expected return of policy μ may be written in terms of state-action pairs (instead of in terms of trajectories as in Eq. 2.1) as

$$\eta(\mu) = \int_{\mathcal{Z}} dz \pi^{\mu}(z) \bar{r}(z), \tag{5.28}$$

where $z = (s, a)$ is a state-action pair and $\pi^{\mu}(z) = \sum_{t=0}^{\infty} \gamma^t P_t^{\mu}(z)$ is a discounted weighting of state-action pairs encountered while following policy μ. In the definition of π^{μ}, the term $P_t^{\mu}(z_t)$ is the t-step state-action occupancy density of policy μ given by

$$P_t^{\mu}(z_t) = \int_{\mathcal{Z}^t} dz_0 \dots dz_{t-1} P_0^{\mu}(z_0) \prod_{i=1}^{t} P^{\mu}(z_i | z_{i-1}).$$

Integrating a out of $\pi^{\mu}(z) = \pi^{\mu}(s, a)$ results in the corresponding discounted weighting of states encountered by following policy μ: $\nu^{\mu}(s) = \int_{\mathcal{A}} da \pi^{\mu}(s, a)$. Unlike ν^{μ} and π^{μ}, $(1 - \gamma)\nu^{\mu}$ and $(1 - \gamma)\pi^{\mu}$

are distributions and are analogous to the stationary distributions over states and state-action pairs of policy μ in the undiscounted setting, respectively.

The policy gradient theorem ([Marbach, 1998, Proposition 1]; [Sutton et al., 2000, Theorem 1]; [Konda and Tsitsiklis, 2000, Theorem 1]) states that the gradient of the expected return, defined by Eq. 5.28, is given by

$$\nabla\eta(\boldsymbol{\theta}) = \int dsda\ \nu(s;\boldsymbol{\theta})\nabla\mu(a|s;\boldsymbol{\theta})Q(s,a;\boldsymbol{\theta}). \tag{5.29}$$

In an AC algorithm, the actor updates the policy parameters $\boldsymbol{\theta}$ along the direction of an estimate of the gradient of the performance measure (Eq. 5.29), while the critic helps the actor in this update by providing it with an estimate of the action-value function $Q(s,a;\boldsymbol{\theta})$. In most existing AC algorithms (both conventional and natural), the actor uses Monte-Carlo (MC) techniques to estimate the gradient of the performance measure and the critic approximates the action-value function using some form of temporal difference (TD) learning [Sutton, 1988].

The idea of Bayesian actor-critic (BAC) [Ghavamzadeh and Engel, 2007] is to apply the Bayesian quadrature (BQ) machinery described in §5.2.1 to the policy gradient expression given by Eq. 5.29 in order to reduce the variance in the gradient estimation procedure (see Figure 5.3 for the connection between the BQ machinery and BAC).

Similar to the BPG methods described in §5.2.2, in BAC, we place a Gaussian process (GP) prior over action-value functions using a prior covariance kernel defined on state-action pairs: $k(z,z') = \mathbf{Cov}\big[Q(z),Q(z')\big]$. We then compute the GP posterior conditioned on the sequence of individual observed transitions. By an appropriate choice of a prior on action-value functions, we are able to derive closed-form expressions for the posterior moments of $\nabla\eta(\boldsymbol{\theta})$. The main questions here are: **1)** how to compute the GP posterior of the action-value function given a sequence of observed transitions? and **2)** how to choose a prior for the action-value function that allows us to derive closed-form expressions for the posterior moments of $\nabla\eta(\boldsymbol{\theta})$? The Gaussian process temporal difference (GPTD) method [Engel et al., 2005a] described in §5.1.1 provides a machinery for computing the posterior moments of

$Q(z)$. After t time-steps, GPTD gives us the following posterior moments for Q (see Eqs. 5.15 and 5.16 in §5.1.1):

$$\widehat{Q}_t(z) = \mathbf{E}\left[Q(z)|\mathcal{D}_t\right] = \boldsymbol{k}_t(z)^\top \boldsymbol{\alpha}_t,$$
$$\widehat{S}_t(z,z') = \mathbf{Cov}\left[Q(z), Q(z')|\mathcal{D}_t\right] = k(z,z') - \boldsymbol{k}_t(z)^\top \boldsymbol{C}_t \boldsymbol{k}_t(z'),$$

where $\mathcal{D}_t$ denotes the observed data up to and including time step t, and

$$\boldsymbol{k}_t(z) = \big(k(z_0,z),\ldots,k(z_t,z)\big)^\top, \quad \boldsymbol{K}_t = \big[\boldsymbol{k}_t(z_0), \boldsymbol{k}_t(z_1), \ldots, \boldsymbol{k}_t(z_t)\big],$$
$$\boldsymbol{\alpha}_t = \boldsymbol{H}_t^\top \Big(\boldsymbol{H}_t \boldsymbol{K}_t \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_t\Big)^{-1} \boldsymbol{r}_{t-1}\ , \quad \boldsymbol{C}_t = \boldsymbol{H}_t^\top \Big(\boldsymbol{H}_t \boldsymbol{K}_t \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_t\Big)^{-1} \boldsymbol{H}_t\ . \tag{5.30}$$

Using the above equations for the posterior moments of Q, making use of the linearity of Eq. 5.29 in Q, and denoting $g(z;\boldsymbol{\theta}) = \pi^\mu(z)\nabla \log \mu(a|s;\boldsymbol{\theta})$, the posterior moments of the policy gradient $\nabla\eta(\boldsymbol{\theta})$ may be written as [O'Hagan, 1991]

$$\mathbf{E}\big[\nabla\eta(\boldsymbol{\theta})|\mathcal{D}_t\big] = \int_{\mathcal{Z}} dz g(z;\boldsymbol{\theta}) \boldsymbol{k}_t(z)^\top \boldsymbol{\alpha}_t,$$
$$\mathbf{Cov}\big[\nabla\eta(\boldsymbol{\theta})|\mathcal{D}_t\big] = \int_{\mathcal{Z}^2} dz dz' g(z;\boldsymbol{\theta}) \left(k(z,z') - \boldsymbol{k}_t(z)^\top \boldsymbol{C}_t \boldsymbol{k}_t(z')\right) g(z';\boldsymbol{\theta})^\top.$$

These equations provide us with the general form of the posterior policy gradient moments. We are now left with a computational issue, namely, how to compute the integrals appearing in these expressions? We need to be able to evaluate the following integrals:

$$\boldsymbol{B}_t = \int_{\mathcal{Z}} dz g(z;\boldsymbol{\theta}) \boldsymbol{k}_t(z)^\top, \qquad \boldsymbol{B}_0 = \int_{\mathcal{Z}^2} dz dz' g(z;\boldsymbol{\theta}) k(z,z') g(z';\boldsymbol{\theta})^\top. \tag{5.31}$$

Using the definitions of $\boldsymbol{B}_t$ and $\boldsymbol{B}_0$, the gradient posterior moments may be written as

$$\mathbf{E}\big[\nabla\eta(\boldsymbol{\theta})|\mathcal{D}_t\big] = \boldsymbol{B}_t \boldsymbol{\alpha}_t\ , \qquad \mathbf{Cov}\big[\nabla\eta(\boldsymbol{\theta})|\mathcal{D}_t\big] = \boldsymbol{B}_0 - \boldsymbol{B}_t \boldsymbol{C}_t \boldsymbol{B}_t^\top. \tag{5.32}$$

In order to render these integrals analytically tractable, Ghavamzadeh and Engel [Ghavamzadeh and Engel, 2007] chose the prior covariance kernel to be the sum of an arbitrary state-kernel k_s

and the Fisher kernel k_F between state-action pairs, i.e.,

$$k_F(z, z') = \boldsymbol{u}(z; \boldsymbol{\theta})^\top \boldsymbol{G}(\boldsymbol{\theta})^{-1} \boldsymbol{u}(z') \;, \qquad k(z, z') = k_s(s, s') + k_F(z, z') \;, \tag{5.33}$$

where $\boldsymbol{u}(z; \boldsymbol{\theta})$ and $\boldsymbol{G}(\boldsymbol{\theta})$ are respectively the score function and the Fisher information matrix, defined as[6]

$$\begin{aligned} \boldsymbol{u}(z; \boldsymbol{\theta}) &= \nabla \log \mu(a|s; \boldsymbol{\theta}), \\ \boldsymbol{G}(\boldsymbol{\theta}) &= \mathbf{E}_{s\sim\nu^\mu, a\sim\mu} \left[\nabla \log \mu(a|s; \boldsymbol{\theta}) \nabla \log \mu(a|s; \boldsymbol{\theta})^\top \right] \\ &= \mathbf{E}_{z\sim\pi^\mu} \left[\boldsymbol{u}(z; \boldsymbol{\theta}) \boldsymbol{u}(z; \boldsymbol{\theta})^\top \right]. \end{aligned} \tag{5.34}$$

Using the prior covariance kernel of Eq. 5.33, Ghavamzadeh and Engel [Ghavamzadeh and Engel, 2007] showed that the integrals in Eq. 5.31 can be computed as

$$\boldsymbol{B}_t = \boldsymbol{U}_t \;, \qquad\qquad \boldsymbol{B}_0 = \boldsymbol{G}, \tag{5.35}$$

where $\boldsymbol{U}_t = \big[\boldsymbol{u}(z_0), \boldsymbol{u}(z_1), \ldots, \boldsymbol{u}(z_t)\big]$. As a result, the integrals of the gradient posterior moments (Eq. 5.32) are analytically tractable (see Ghavamzadeh et al. [2013] for more details). An immediate consequence of Eq. 5.35 is that, in order to compute the posterior moments of the policy gradient, we only need to be able to evaluate (or estimate) the score vectors $\boldsymbol{u}(z_i),\ i = 0, \ldots, t$ and the Fisher information matrix $\boldsymbol{G}$ of the policy.

Similar to the BPG method, Ghavamzadeh and Engel [Ghavamzadeh and Engel, 2007] suggest methods for online estimation of the Fisher information matrix in Eq. 5.34 and for using online sparsification to make the BAC algorithm more efficient in both time and memory (see Ghavamzadeh et al. [2013] for more details). They also report experimental results [Ghavamzadeh and Engel, 2007, Ghavamzadeh et al., 2013] which indicate that the BAC algorithm tends to significantly reduce the number of samples needed to obtain accurate gradient estimates, and thus, given a fixed number of samples per iteration, finds a better policy than both MC-based policy gradient

[6]Similar to $\boldsymbol{u}(\xi)$ and $\boldsymbol{G}$ defined by Eqs. 5.19 and 5.27, to simplify the notation, we omit the dependence of $\boldsymbol{u}$ and $\boldsymbol{G}$ to the policy parameters $\boldsymbol{\theta}$, and replace $\boldsymbol{u}(z; \boldsymbol{\theta})$ and $\boldsymbol{G}(\boldsymbol{\theta})$ with $\boldsymbol{u}(z)$ and $\boldsymbol{G}$ in the sequel.

methods and the BPG algorithm, which do not take into account the Markovian property of the system.

6

Risk-aware Bayesian Reinforcement Learning

The results presented so far have all been concerned with optimizing the *expected return* of the policy. However, in many applications, the decision-maker is also interested in minimizing the *risk* of a policy. By risk,[1] we mean performance criteria that take into account not only the expected return, but also some additional statistics of it, such as variance, Value-at-Risk (VaR), expected shortfall (also known as conditional-value-at-risk or CVaR), etc. The primary motivation for dealing with risk-sensitive performance criteria comes from finance, where risk plays a dominant role in decision-making. However, there are many other application domains where risk is important, such as process control, resource allocation, and clinical decision-making.

In general, there are two sources that contribute to the reward uncertainty in MDPs: *internal uncertainty* and *parametric uncertainty*. Internal uncertainty reflects the uncertainty of the return due to the stochastic transitions and rewards, for a single and *known* MDP. Parametric uncertainty, on the other hand, reflects the uncertainty about the unknown MDP parameters – the transition and reward *distribu-*

[1] The term risk here should not be confused with the Bayes risk, defined in Chapter 3.

tions. As a concrete example of this dichotomy consider the following two experiments. First, select a *single* MDP and execute some fixed policy on it several times. In each execution, the return may be different due to the stochastic transitions and reward. This variability corresponds to the internal uncertainty. In the second experiment, consider drawing several MDPs from some distribution (typically, the posterior distribution in a Bayesian setting). For each drawn MDP, execute the same policy several times and compute the average return across the executions. The variability in the *average* returns across the different MDPs corresponds to the parametric type of uncertainty. The Bayesian setting offers a framework for dealing with parameter uncertainty in a principled manner. Therefore, work on risk-sensitive RL in the Bayesian setting focuses on risk due to parametric uncertainty, as we shall now survey.

Bias and Variance Approximation in Value Function Estimates

The first result we discuss concerns policy evaluation. Mannor et al. [Mannor et al., 2007] derive approximations to the bias and variance of the value function of a fixed policy due to parametric uncertainty.

Consider an MDP $\mathcal{M}$ with *unknown* transition probabilities P,[2] a Dirichlet prior on the transitions, and assume that we have observed $n(s, a, s')$ transitions from state s to state s' under action a. Recall that the posterior transition probabilities are also Dirichlet, and may be calculated as outlined in §2.5.1.[3] Consider also a stationary Markov policy μ, and let P^μ denote the unknown transition probabilities induced by μ in the unknown MDP $\mathcal{M}$ and V^μ denote the corresponding value function. Recall that V^μ is the solution to the Bellman equation (2.4) and may be written explicitly as $V^\mu = (I - \gamma P^\mu)^{-1} R^\mu$, where we recall that R^μ denotes the expected rewards induced by μ. The Bayesian formalism allows the calculation of the *posterior* mean and covariance of V^μ, given the observations.

[2]Following the framework of Chapter 4, we assume that the rewards are known and only the transitions are unknown. The following results may be extended to cases where the rewards are also unknown, as outlined in §4.7.

[3]Note that this is similar to the BAMDP formulation of §4.1, where the hyperstate encodes the posterior probabilities of the transitions given the observations.

Let $\hat{P}(s'|s,a) = \mathbf{E}_{\text{post}}[P(s'|s,a)]$ denote the expected transition probabilities with respect to their posterior distribution, and let $\hat{P}^{\mu}(s'|s) = \sum_a \mu(a|s)\hat{P}(s'|s,a)$ denote the expected transitions induced by μ. We denote by $\hat{V}^{\mu} = (I - \gamma\hat{P}^{\mu})^{-1}R^{\mu}$ the *estimated* value function. Also, let $\mathbf{e}$ denote a vector of ones. The posterior mean and covariance of V^{μ} are given in the following two theorems:

Theorem 6.1. [Mannor et al., 2007] The expectation (under the posterior) of V^{μ} satisfies

$$\mathbf{E}_{\text{post}}[V^{\mu}] = \hat{V}^{\mu} + \gamma^2\hat{X}\hat{Q}\hat{V}^{\mu} + \gamma\hat{B} + L_{\text{bias}},$$

where $\hat{X} = \left(I - \gamma\hat{P}^{\mu}\right)^{-1}$, and the vector $\hat{B}$ and matrix $\hat{Q}$ are computed according to

$$\hat{B}_i = \sum_a \mu(a|i)^2 R(i,a)\mathbf{e}^{\top}\hat{M}^{i,a}\hat{X}_{\cdot,i},$$

and

$$\hat{Q}_{i,j} = \hat{\mathbf{Cov}}^{(i)}_{j,\cdot}\hat{X}_{\cdot,i} \qquad \text{in which} \qquad \hat{\mathbf{Cov}}^{(i)} = \sum_a \mu(a|i)^2\hat{M}^{i,a},$$

where the matrix $\hat{M}^{i,a}$ is the posterior covariance matrix of $P(\cdot|s,a)$, and higher order terms

$$L_{\text{bias}} = \sum_{k=3}^{\infty}\gamma^k\mathbf{E}\left[f_k(\tilde{P})\right]R^{\mu},$$

in which $\tilde{P} = P - \hat{P}$, and $f_k(\tilde{P}) = \hat{X}\left(\tilde{P}\hat{X}\right)^k$.

Theorem 6.2. [Mannor et al., 2007] Using the same notations as Theorem 6.1, the second moment (under the posterior) of V^{μ} satisfies

$$\begin{aligned}\mathbf{E}_{\text{post}}[V^{\mu}V^{\mu\top}] = \hat{V}^{\mu}\hat{V}^{\mu^{\top}} + \hat{X}\Big(&\gamma^2\big(\hat{Q}\hat{V}^{\mu}R^{\mu\top} + R^{\mu}\hat{V}^{\mu^{\top}}\hat{Q}^{\top}\big)\\ &+ \gamma\big(\hat{B}R^{\mu\top} + R^{\mu}\hat{B}^{\top}\big) + \hat{W}\Big)\hat{X}^{\top} + L_{\text{var}},\end{aligned}$$

where $\hat{W}$ is a diagonal matrix with elements

$$\hat{W}_{i,i} = \sum_a \mu(a|i)^2(\gamma V^{\mu^{\top}} + R(i,a)\mathbf{e})\hat{M}^{i,a}(\gamma V^{\mu} + R(i,a)\mathbf{e}^{\top}),$$

and higher order terms

$$L_{\text{var}} = \sum_{k,l:k+l>2} \gamma^{k+l} \mathbf{E}\left[f_k(\tilde{P}) R^{\mu} R^{\mu\top} f_l(\tilde{P})^{\top} \right].$$

It is important to note that except for the higher order terms L_{bias} and L_{var}, the terms in Theorems 6.1 and 6.2 do not depend on the unknown transitions, and thus, may be calculated *from the data*. This results in a second-order approximation of the bias and variance of V^{μ}, which may be used to derive confidence intervals around the estimated value function. This approximation was also used by Delage and Mannor [Delage and Mannor, 2010] for risk-sensitive policy optimization, as we describe next.

Percentile Criterion

Consider again a setting where $n(s, a, s')$ transitions from state s to state s' under action a from MDP $\mathcal{M}$ were observed, and let P_{post} denote the posterior distribution of the transition probabilities in $\mathcal{M}$ given these observations. Delage and Mannor [Delage and Mannor, 2010] investigate the *percentile criterion*[4] for $\mathcal{M}$, defined as

$$\begin{aligned} \max_{y \in \mathbb{R}, \mu} \quad & y \\ \text{s.t.} \quad & P_{\text{post}}\left(\mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(Z_t) \,\middle|\, Z_0 \sim P_0, \mu \right] \geq y \right) \geq 1 - \epsilon, \end{aligned} \tag{6.1}$$

where P_{post} denotes the probability of drawing a transition matrix P' from the posterior distribution of the transitions, and the expectation is with respect to a concrete realization of that P'. Note that the value of the optimization problem in (6.1) is a $(1-\epsilon)$-guarantee to the performance of the optimal policy with respect to the parametric uncertainty.

Unfortunately, solving (6.1) for general uncertainty in the parameters is NP-hard [Delage and Mannor, 2010]. However, for the case of a Dirichlet prior, the 2nd order approximation of Theorem 6.1 may be used to derive an approximately optimal solution.

[4]This is similar to the popular financial risk measure Value-at-Risk (VaR). However, note that VaR is typically used in the context of *internal* uncertainty.

Let $F(\mu)$ denote the 2nd order approximation[5] of the expected return under policy μ and initial state distribution P_0 (cf. Theorem 6.1)

$$F(\mu) = P_0^\top \hat{V}^\mu + \gamma^2 P_0^\top \hat{X}\hat{Q}\hat{V}^\mu.$$

The next result of [Delage and Mannor, 2010] shows that given enough observations, optimizing $F(\mu)$ leads to an approximate solution of the percentile problem (6.1).

Theorem 6.3. [Delage and Mannor, 2010] Let $N^* = \min_{s,a} \sum_{s'} n(s,a,s')$ denote the minimum number of state transitions observed from any state using any action, and $\epsilon \in (0, 0.5]$. The policy

$$\hat{\mu} = \arg\max_{\mu} F(\mu)$$

is $\mathcal{O}(1/\sqrt{\epsilon N^*})$ optimal with respect to the percentile optimization criterion (6.1).

Note that, as discussed earlier, $F(\mu)$ may be efficiently evaluated for every μ. However, $F(\mu)$ is non-convex in μ, but empirically, global optimization techniques for maximizing $F(\mu)$ lead to useful solutions [Delage and Mannor, 2010].

Delage and Mannor [Delage and Mannor, 2010] also consider a case where the state transitions are known, but there is uncertainty in the reward distribution. For general reward distributions the corresponding percentile optimization problem is also NP-hard. However, for the case of a Gaussian prior, the resulting optimization problem is a second-order cone program, for which efficient solutions are known.

Max-Min Criterion

Consider the percentile criterion (6.1) in the limit of $\epsilon \to 0$. In this case, we are interested in the performance under the worst realizable posterior transition probability, i.e.,

$$\max_{\mu} \min_{P \in \mathcal{P}_{\text{post}}} \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(Z_t) \,\middle|\, Z_0 \sim P_0, P, \mu \right], \tag{6.2}$$

[5]Note that the 1st order term is ignored, since it would cancel anyway in the optimization that follows.

where $\mathcal{P}_{\text{post}}$ denotes the set of all realizable transition probabilities in the posterior. For the case of a Dirichlet prior, this criterion is useless, as the set $\mathcal{P}_{\text{post}}$ contains the entire simplex for each state. Bertuccelli et al. [Bertuccelli et al., 2012] consider instead a *finite* subset[6] $\hat{\mathcal{P}}_{\text{post}} \in \mathcal{P}_{\text{post}}$, and minimize only over the set $\hat{\mathcal{P}}_{\text{post}}$, resulting in the following criterion:

$$\max_{\mu} \min_{P \in \hat{\mathcal{P}}_{\text{post}}} \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(Z_t) \,\middle|\, Z_0 \sim P_0, P, \mu\right]. \tag{6.3}$$

Given a set $\hat{\mathcal{P}}_{\text{post}}$, the optimization in (6.3) may be solved efficiently using min-max dynamic programming. Thus, the 'real question' is how to construct $\hat{\mathcal{P}}_{\text{post}}$. Naturally, the construction of $\hat{\mathcal{P}}_{\text{post}}$ should reflect the posterior distribution of transition probabilities. One approach is to construct it from a finite number of models sampled from the posterior distribution, reminiscent of Thompson sampling. However, as claimed in [Bertuccelli et al., 2012], this approach requires a very large number of samples in order to adequately represent the posterior.

Alternatively, Bertuccelli et al. [Bertuccelli et al., 2012] propose a deterministic sampling procedure for the Dirichlet distribution based on sigma-points. In simple terms, sigma-points are points placed around the posterior mean and spaced proportionally to the posterior variance. The iterative algorithm of [Bertuccelli et al., 2012] consists of two phases. In the first phase, the already observed transitions are used to derive the posterior Dirichlet distribution, for which an optimal policy is derived by solving (6.3). In the second phase, this policy is then acted upon in the system to generate additional observations. As more data become available, the posterior variance decreases and the sigma points become closer, leading to convergence of the algorithm.

Percentile Measures Criteria

The NP-hardness of the percentile criterion for general uncertainty structures motivates a search for more tractable risk-aware performance criteria. Chen and Bowling [Chen and Bowling, 2012] propose to replace the individual percentile with a *measure* over percentiles. For-

[6] This is also known as a set of *scenarios*.

mally, given a measure ψ over the interval $[0, 1]$, consider the following optimization problem:

$$\begin{aligned} \max_{\mu, y \in \mathcal{F}} \quad & \int_x y(x) d\psi \\ \text{s.t.} \quad & P_{\text{post}} \left(\mathbf{E} \left[\sum_{t=0}^{T} R(Z_t) \,\middle|\, Z_0 \sim P_0, \mu \right] \geq y(x) \right) \geq x \qquad \forall x \in [0, 1], \end{aligned} \tag{6.4}$$

where $\mathcal{F}$ is the class of real-valued and bounded ψ-integrable functions on the interval $[0, 1]$, P_{post} denotes the probability of drawing a transition matrix P' from the posterior distribution of the transitions (as before), and the expectation is with respect to a concrete realization of that P'. Note that here the horizon is T and there is no discounting, as opposed to the infinite horizon discounted setting discussed earlier.

The percentile measure criterion (6.4) may be seen as a generalization of the percentile criterion (6.1), which is obtained by setting ψ to a Dirac delta at $1 - \epsilon$. In addition, when ψ is uniform on $[0, 1]$, (6.4) is equivalent to the expected value of Theorem 6.1, and when ψ is a delta Dirac at 0, we obtain the max-min criterion (6.2). Finally, when ψ is a step function, an expected shortfall (CVaR) criterion is obtained.

Chen and Bowling [Chen and Bowling, 2012] introduce the k-of-N family of percentile measures, which admits an efficient solution under a general uncertainty structure. For a policy μ, the k-of-N measure is equivalent to the following sampling procedure: first draw N MDPs according to the posterior distribution, then select a set of the k MDPs with the worst expected performance under policy μ, and finally choose a random MDP from this set (according to a uniform distribution). By selecting suitable k and N, the k-of-N measure may be tuned to closely approximate the CVaR or max-min criterion.

The main reason for using the k-of-N measure is that the above sampling procedure may be seen as a two-player zero-sum extensive-form game with imperfect information, which may be solved efficiently using counterfactual regret minimization. This results in the following convergence guarantee:

Theorem 6.4. [Chen and Bowling, 2012] For any $\epsilon > 0$ and $\delta \in (0, 1]$,

let

$$\bar{T} = \left(1 + \frac{2}{\sqrt{\delta}}\right)^2 \frac{16T^2\Delta^2|\mathcal{I}_1|^2|\mathcal{A}|}{\delta^2\epsilon^2},$$

where $T\Delta$ is the maximum difference in total reward over T steps. With probability $1-\delta$, the current strategy at iteration T^*, chosen uniformly at random from the interval $[1, \bar{T}]$, is an ϵ-approximation to a solution of (6.4) when ψ is a k-of-N measure. The total time complexity is $\mathcal{O}\left((T\Delta/\epsilon)^2 \frac{|\mathcal{I}_1|^3|\mathcal{A}|^3 N \log N}{\delta^3}\right)$, where $|\mathcal{I}_1| \in \mathcal{O}\left(|\mathcal{S}|T\right)$ for arbitrary reward uncertainty and $|\mathcal{I}_1| \in \mathcal{O}\left(|\mathcal{S}|^{T+1}\mathcal{A}^T\right)$ for arbitrary transition and reward uncertainty.

The exponential dependence on the horizon T in Theorem 6.4 is due to the fact that an optimal policy for the risk-sensitive criterion (6.4) is not necessarily Markov and may depend on the complete history. In comparison, the previous results avoided this complication by searching only in the space of Markov policies. An interesting question is whether other choices of measure ψ admit an efficient solution. Chen and Bowling [Chen and Bowling, 2012] provide the following sufficient condition for tractability:

Theorem 6.5. [Chen and Bowling, 2012] Let ψ be an absolutely continuous measure with density function g_ψ, such that g_ψ is non-increasing and piecewise Lipschitz continuous with m pieces and Lipschitz constant L. A solution of (6.4) can be approximated with high probability in time polynomial in $\left\{|\mathcal{A}|, |\mathcal{S}|, \Delta, L, m, \frac{1}{\epsilon}, \frac{1}{\delta}\right\}$ for (i) arbitrary reward uncertainty with time also polynomial in the horizon or (ii) arbitrary transition and reward uncertainty with a fixed horizon.

Note that in line with the previous hardness results, both the CVaR and max-min criteria may be represented using a non-increasing and piecewise Lipschitz continuous measure, while the percentile criterion may not.

7

BRL Extensions

In this section, we discuss extensions of the Bayesian reinforcement learning (BRL) framework to the following classes of problems: PAC-Bayes model selection, inverse RL, multi-agent RL, and multi-task RL.

7.1 PAC-Bayes Model Selection

While Bayesian RL provides a rich framework for incorporating domain knowledge, one of the often mentioned limitations is the requirement to have *correct* priors, meaning that the prior has to admit the true posterior. Of course this issue is pervasive across Bayesian learning methods, not just Bayesian RL. Recent work on PAC-Bayesian analysis seeks to provide tools that are robust to poorly selected priors. PAC-Bayesian methods provide a way to simultaneously exploit prior knowledge when it is appropriate, while providing distribution-free guarantees based on properties such as VC dimension [McAllester, 1999].

PAC-Bayesian bounds for RL in finite state spaces were introduced by [Fard and Pineau, 2010], showing that it is possible to remove the assumption on the correctness of the prior, and instead, measure the consistency of the prior over the training data. Bounds are available for both model-based RL, where a prior distribution is given on the space of possible models, and model-free RL, where a prior is given on

the space of value functions. In both cases, the bound depends on an empirical estimate and a measure of distance between the stochastic policy and the one imposed by the prior distribution. The primary use of these bounds is for model selection, where the bounds can be useful in choosing between following the empirical estimate and the Bayesian posterior, depending on whether the prior was informative or misleading. PAC-Bayesian bounds for the case of RL with function approximation are also available to handle problems with continuous state spaces [Fard et al., 2011].

For the most part, PAC-Bayesian analysis to date has been primarily theoretical, with few empirical results. However, recent theoretical results on PAC-Bayesian analysis of the bandit case may provide useful tools for further development of this area [Seldin et al., 2011a,b].

7.2 Bayesian Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) is the problem of learning the underlying model of the decision-making agent (expert) from its observed behavior and the dynamics of the system [Russell, 1998]. IRL is motivated by situations in which the goal is only to learn the reward function (as in preference elicitation) and by problems in which the main objective is to learn good policies from the expert (apprenticeship learning). Both reward learning (direct) and apprenticeship learning (indirect) views of this problem have been studied in the last decade (e.g., [Ng and Russell, 2000, Abbeel and Ng, 2004, Ratliff et al., 2006, Neu and Szepesvári, 2007, Ziebart et al., 2008, Syed and Schapire, 2008]). What is important is that the IRL problem is inherently *ill-posed* since there might be an infinite number of reward functions for which the expert's policy is optimal. One of the main differences between the various works in this area is in how they formulate the reward preference in order to obtain a *unique* reward function for the expert.

The main idea of Bayesian IRL (BIRL) is to use a prior to encode the reward preference and to formulate the compatibility with the expert's policy as a likelihood in order to derive a probability distribution over the space of reward functions, from which the expert's

reward function is somehow extracted. Ramachandran and Amir [Ramachandran and Amir, 2007] use this BIRL formulation and propose a Markov chain Monte Carlo (MCMC) algorithm to find the posterior mean of the reward function and return it as the reward of the expert. Michini and How [Michini and How, 2012b] improve the efficiency of the method in [Ramachandran and Amir, 2007] by not including the entire state space in the BIRL inference. They use a kernel function that quantifies the similarity between states and scales down the BIRL inference by only including those states that are similar (the similarity is defined by the kernel function) to the ones encountered by the expert. Choi and Kim [Choi and Kim, 2011] use the BIRL formulation of [Ramachandran and Amir, 2007] and first show that using the posterior mean may not be a good idea since it may yield a reward function whose corresponding optimal policy is inconsistent with the expert's behaviour; the posterior mean integrates the error over the entire space of reward functions by including (possibly) infinitely many rewards that induce policies that are inconsistent with the expert's demonstration. Instead, they suggest that the maximum-a-posteriori (MAP) estimate could be a better solution for IRL. They formulate IRL as a posterior optimization problem and propose a gradient method to calculate the MAP estimate that is based on the (sub)differentiability of the posterior distribution. Finally, they show that most of the non-Bayesian IRL algorithms in the literature [Ng and Russell, 2000, Ratliff et al., 2006, Neu and Szepesvári, 2007, Ziebart et al., 2008, Syed and Schapire, 2008] can be cast as searching for the MAP reward function in BIRL with different priors and different ways of encoding the compatibility with the expert's policy.

Using a single reward function to explain the expert's behavior might be problematic as its complexity grows with the complexity of the task being optimized by the expert. Searching for a complex reward function is often difficult. If many parameters are needed to model it, we are required to search over a large space of candidate functions. This problem becomes more severe when we take into account the testing of each candidate reward function, which requires solving an MDP for its optimal value function, whose computation usually scales poorly with

the size of the state space. Michini and How [Michini and How, 2012a] suggest a potential solution to this problem in which they partition the observations (system trajectories generated by the expert) into sets of smaller sub-demonstrations, such that each sub-demonstration is attributed to a smaller and less-complex class of reward functions. These simple rewards can be intuitively interpreted as *sub-goals* of the expert. They propose a BIRL algorithm that uses a Bayesian non-parametric mixture model to automate this partitioning process. The proposed algorithm uses a Chinese restaurant process prior over partitions so that there is no need to specify the number of partitions a priori.

Most of the work in IRL assumes that the data is generated by a single expert optimizing a fixed reward function. However, there are many applications in which we observe multiple experts, each executing a policy that is optimal (or good) with respect to its own reward function. There have been a few works that consider this scenario. Dimitrakakis and Rothkopf [Dimitrakakis and Rothkopf, 2011] generalize BIRL to multi-task learning. They assume a common prior for the trajectories and estimate the reward functions individually for each trajectory. Other than the common prior, they do not make any additional efforts to group trajectories that are likely to be generated from the same or similar reward functions. Babes et al. [Babes et al., 2011] propose a method that combines expectation maximization (EM) clustering with IRL. They cluster the trajectories based on the inferred reward functions, where one reward function is defined per cluster. However, their proposed method is based on the assumption that the number of clusters (the number of the reward functions) is known. Finally, Choi and Kim [Choi and Kim, 2012] present a nonparametric Bayesian approach using the Dirichlet process mixture model in order to address the IRL problem with multiple reward functions. They develop an efficient Metropolis-Hastings sampler that utilizes the gradient of the reward function posterior to infer the reward functions from the behavior data. Moreover, after completing IRL on the behavior data, their method can efficiently estimate the reward function for a new trajectory by computing the mean of the reward function posterior, given the pre-learned results.

7.3 Bayesian Multi-agent Reinforcement Learning

There is a rich literature on the use of RL methods for multi-agent systems. More recently, the extension of BRL methods to collaborative multi-agent systems has been proposed [Chalkiadakis and Boutilier, 2013]. The objective in this case is consistent with the standard BRL objective, namely to optimally balance the cost of exploring the world with the expected benefit of new information. However, when dealing with multi-agent systems, the complexity of the decision problem is increased in the following way: while single-agent BRL requires maintaining a posterior over the MDP parameters (in the case of model-based methods) or over the value/policy (in the case of model-free methods), in multi-agent BRL, it is also necessary to keep a posterior over the policies of the other agents. This belief can be maintained in a tractable manner subject to certain structural assumptions on the domain, for example that the strategies of the agents are independent of each other. Alternatively, this framework can be used to control the formation of coalitions among agents [Chalkiadakis et al., 2010]. In this case, the Bayesian posterior can be limited to the uncertainty about the capabilities of the other agents, and the uncertainty about the effects of coalition actions among the agents. The solution to the Bayes-optimal strategy can be approximated using a number of techniques, including myopic or short (1-step) lookahead on the value function, or an extension of the value-of-information criteria proposed by [Dearden et al., 1998].

7.4 Bayesian Multi-Task Reinforcement Learning

Multi-task learning (MTL) is an important learning paradigm and active area of research in machine learning (e.g., [Caruana, 1997, Baxter, 2000]). A common setup in MRL considers multiple related tasks for which we are interested in improving the performance of individual learning by sharing information across tasks. This transfer of information is particularly important when we are provided with only a limited number of data with which learn each task. Exploiting data from related problems provides more training samples for the learner and can

improve the performance of the resulting solution. More formally, the main objective in MTL is to maximize the improvement over individual learning, averaged over multiple tasks. This should be distinguished from transfer learning in which the goal is to learn a suitable bias for a class of tasks in order to maximize the expected future performance.

Traditional RL algorithms typically do not directly take advantage of information coming from other similar tasks. However recent work has shown that transfer and multi-task learning techniques can be employed in RL to reduce the number of samples needed to achieve nearly-optimal solutions. All approaches to multi-task RL (MTRL) assume that the tasks share similarity in some components of the problem such as dynamics, reward structure, or value function. While some methods explicitly assume that the shared components are drawn from a common generative model [Wilson et al., 2007, Mehta et al., 2008, Lazaric and Ghavamzadeh, 2010], this assumption is more implicit in others. In [Mehta et al., 2008], tasks share the same dynamics and reward features, and only differ in the weights of the reward function. The proposed method initializes the value function for a new task using the previously learned value functions as a prior. Wilson et al. [Wilson et al., 2007] and Lazaric and Ghavamzadeh [Lazaric and Ghavamzadeh, 2010] both assume that the distribution over some components of the tasks is drawn from a hierarchical Bayesian model (HBM). We describe these two methods in more detail below.

Lazaric and Ghavamzadeh [Lazaric and Ghavamzadeh, 2010] study the MTRL scenario in which the learner is provided with a number of MDPs with common state and action spaces. For any given policy, only a small number of samples can be generated in each MDP, which may not be enough to accurately evaluate the policy. In such a MTRL problem, it is necessary to identify classes of tasks with similar structure and to learn them jointly. It is important to note that here a task is a pair of MDP and policy (i.e., a Markov chain) such that all the MDPs have the same state and action spaces. They consider a particular class of MTRL problems in which the tasks *share structure in their value functions*. To allow the value functions to share a common structure, it is assumed that they are all sampled from a common prior. They

adopt the GPTD value function model ([Engel et al., 2005a], also see §5.1.1) for each task, model the distribution over the value functions using an HBM, and develop solutions to the following problems: (i) joint learning of the value functions (multi-task learning), and (ii) efficient transfer of the information acquired in (i) to facilitate learning the value function of a newly observed task (transfer learning). They first present an HBM for the case in which all of the value functions belong to the same class, and derive an EM algorithm to find MAP estimates of the value functions and the model's hyper-parameters. However, if the functions do not belong to the same class, simply learning them together can be detrimental (*negative transfer*). It is therefore important to have models that will generally benefit from related tasks and will not hurt performance when the tasks are unrelated. This is particularly important in RL as changing the policy at each step of policy iteration (this is true even for fitted value iteration) can change the way in which tasks are clustered together. This means that even if we start with value functions that all belong to the same class, after one iteration the new value functions may be clustered into several classes. To address this issue, they introduce a Dirichlet process (DP) based HBM for the case where the value functions belong to an undefined number of classes, and derive inference algorithms for both the multi-task and transfer learning scenarios in this model.

The MTRL approach in [Wilson et al., 2007] also uses a DP-based HBM to model the distribution over a common structure of the tasks. In this work, the tasks *share structure in their dynamics and reward functions*. The setting is incremental, i.e., the tasks are observed as a sequence, and there is no restriction on the number of samples generated by each task. The focus is not on joint learning with finite number of samples, but on using the information gained from the previous tasks to facilitate learning in a new one. In other words, the focus in this work is on transfer and not on multi-task learning.

8

Outlook

Bayesian reinforcement learning (BRL) offers a coherent probabilistic model for reinforcement learning. It provides a principled framework to express the classic exploration-exploitation dilemma, by keeping an explicit representation of uncertainty, and selecting actions that are optimal with respect to a version of the problem that incorporates this uncertainty. This framework is of course most useful for tackling problems where there is an explicit representation of prior information. Throughout this survey, we have presented several frameworks for leveraging this information, either over the model parameters (model-based BRL) or over the solution (model-free BRL).

In spite of its elegance, BRL has not, to date, been widely applied. While there are a few successful applications of BRL for advertising [Graepel et al., 2010, Tang et al., 2013] and robotics [Engel et al., 2005b, Ross et al., 2008b], adoption of the Bayesian framework in applications is lagging behind the comprehensive theory that BRL has to offer. In the remainder of this section we analyze the perceived limitations of BRL and offer some research directions that might circumvent these limitations.

One perceived limitation of Bayesian RL is the need to provide a

prior. While this is certainly the case for model-based BRL, for larger problems there is always a need for some sort of regularization. A prior serves as a mean to regularize the model selection problem. Thus, the problem of specifying a prior is addressed by *any* RL algorithm that is supposed to work for large-scale problems. We believe that a promising direction for future research concerns devising priors based on observed data, as per the empirical Bayes approach [Efron, 2010]. A related issue is model mis-specification and how to quantify the performance degradation that may arise from not knowing the model precisely.

There are also some algorithmic and theoretical challenges that we would like to point out here. First, scaling BRL is a major issue. Being able to solve large problems is still an elusive goal. We should mention that currently, a principled approach for scaling up RL in general is arguably missing. Approximate value function methods [Bertsekas and Tsitsiklis, 1996] have proved successful for solving certain large scale problems [Powell, 2011], and policy search has been successfully applied to many robotics tasks [Kober et al., 2013]. However, there is currently no solution (neither frequentist nor Bayesian) to the exploration-exploitation dilemma in large-scale MDPs. We hope that scaling up BRL, in which exploration-exploitation is naturally handled, may help us overcome this barrier. Conceptually, BRL may be easier to scale since it allows us, in some sense, to embed domain knowledge into a problem. Second, the BRL framework we presented assumes that the model is specified correctly. Dealing with model misspecification and consequently with model selection is a thorny issue for all RL algorithms. When the state parameters are unknown or not observed, the dynamics may stop being Markov and many RL algorithms fail in theory and in practice. The BRL framework may offer a solution to this issue since the Bayesian approach can naturally handle model selection and misspecification by not committing to a single model and rather sustaining a posterior over the possible models.

Another perceived limitation of BRL is the complexity of *implementing* the majority of BRL methods. Indeed, some frequentist RL algorithms are more elegant than their Bayesian counterparts, which may discourage some practitioners from using BRL. This is, by no

means, a general rule. For example, Thompson sampling is one of the most elegant approaches to the multi-armed bandit problem. Fortunately, the recent release of a software library for Bayesian RL algorithms promises to facilitate this [bbr, 2015, Castronovo et al., 2015]. We believe that Thompson sampling style algorithms (e.g., [Russo and Van Roy, 2014b, Gopalan and Mannor, 2015]) can pave the way to efficient algorithms in terms of both sample complexity and computational complexity. Such algorithms require, essentially, solving an MDP once in a while while taking nearly optimal exploration rate.

From the application perspective, we believe that BRL is still in its infancy. In spite of some early successes, especially for contextual bandit problems [Chapelle and Li, 2011], most of the successes of large-scale real applications of RL are not in the Bayesian setting. We hope that this survey will help facilitate the research needed to make BRL a success in practice. A considerable benefit of BRL is its ability to work well "out-of-the-box": you only need to know relatively little about the uncertainty to perform well. Moreover, adding complicating factors such as long-term constraints or short-term safety requirements can be easily embedded in the framework.

From the modelling perspective, deep learning is becoming increasingly more popular as a building block in RL. It would be interesting to use probabilistic deep networks, such as restricted Boltzman machines as an essential ingredient in BRL. Probabilistic deep models can not only provide a powerful value function or policy approximator, but also allow using historical traces that come from a different problem (as in transfer learning). We believe that the combination of powerful models for value or policy approximation in combination with a BRL approach can further facilitate better exploration policies.

From the conceptual perspective, the main question that we see as open is how to embed domain knowledge into the model? One approach is to modify the prior according to domain knowledge. Another is to consider parametrized or factored models. While these approaches may work well for particular domains, they require careful construction of the state space. However, in some cases, such a careful construction defeats the purpose of a fast and robust "out-of-the-box" approach. We

believe that approaches that use causality and non-linear dimensionality reduction may be the key to using BRL when a careful construction of a model is not feasible. In that way, data-driven algorithms can discover simple structures that can ultimately lead to fast BRL algorithms.

Acknowledgements

The authors extend their warmest thanks to Michael Littman, James Finlay, Melanie Lyman-Abramovitch and the anonymous reviewers for their insights and support throughout the preparation of this manuscript. The authors wish to also thank several colleagues for helpful discussions on this topic: Doina Precup, Amir-massoud Farahmand, Brahim Chaib-draa, and Pascal Poupart. The review of the model-based Bayesian RL approaches benefited significantly from comprehensive reading lists posted online by John Asmuth and Chris Mansley. Funding for Joelle Pineau was provided by the Natural Sciences and Engineering Research Council Canada (NSERC) through their Discovery grants program, by the Fonds de Québécois de Recherche Nature et Technologie (FQRNT) through their Projet de recherche en équipe program. Funding for Shie Mannor and Aviv Tamar were partially provided by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 306638 (SUPREL). Aviv Tamar is also partially funded by the Viterbi Scholarship, Technion.

Appendices

A

Index of Symbols

Here we present a list of the symbols used in this paper to provide a handy reference.[1]

Notation	Definition
$\mathbb{R}$	set of real numbers
$\mathbb{N}$	set of natural numbers
$\mathbf{E}$	expected value
Var	variance
Cov	covariance
KL	Kullback-Leibler divergence
H	Shannon entropy
$\mathcal{N}(m, \sigma^2)$	Gaussian (normal) distribution with mean m and variance σ^2
$\mathcal{P}$	probability distribution
$\mathcal{M}$	model
$\mathcal{A}$	set of actions
K	cardinality of $\mathcal{A}$
T	time horizon
$a \in \mathcal{A}$	a nominal action

[1] In this paper, we use upper-case and lower-case letters to refer to random variables and the values taken by random variables, respectively. We also use bold-face letters for vectors and matrices.

Notation	Definition
$\mathcal{Y}$	set of outcomes (in MAB)
$y \in \mathcal{Y}$	a nominal outcome
$r(y)$	reward for outcome y
$P(y\|a) \in \mathcal{P}(\mathcal{Y})$	probability of observing outcome y after taking action a
a^*	optimal arm
Δ_a	difference in expected reward between arms a and a^*
$\mathcal{S}$	set of states (or contexts in a contextual MAB)
$s \in \mathcal{S}$	a nominal state (context)
$P_S(s) \in \mathcal{P}(\mathcal{S})$	context probability (in contextual MAB)
$P(y\|a, s) \in \mathcal{P}(\mathcal{Y})$	probability of observing outcome y after taking action a when the context is s (in contextual MAB)
$\mathcal{O}$	set of observations
$o \in \mathcal{O}$	a nominal observation
$P \in \mathcal{P}(\mathcal{S})$	transition probability function
$P(s'\|s, a) \in \mathcal{P}(\mathcal{S})$	probability of being in state s' after taking action a in state s
$\Omega(o\|s, a) \in \mathcal{P}(\mathcal{O})$	probability of seeing observation o after taking action a to reach state s
$q \in \mathcal{P}(\mathbb{R})$	probability distribution over reward
$R(s, a) \sim q(\cdot\|s, a)$	random variable of the reward of taking action a in state s
$r(s, a)$	reward of taking action a in state s
$\bar{r}(s, a)$	expected reward of taking action a in state s
$R_{\max}$	maximum random immediate reward
$\bar{R}_{\max}$	maximum expected immediate reward
$P_0 \in \mathcal{P}(\mathcal{S})$	initial state distribution
$b_t \in \mathcal{P}(\mathcal{S})$	POMDP's state distribution at time t
$\tau(b_t, a, o)$	the information state (belief) update equation
μ	a (stationary and Markov) policy
$\mu(a\|s)$	probability that policy μ selects action a in state s
μ^*	an optimal policy
$\mathcal{M}^\mu$	the Markov chain induced by policy μ
P^μ	probability distribution of the Markov chain induced by policy μ
$P^\mu(s'\|s)$	probability of being in state s' after taking an action according to policy μ in state s
P_0^μ	initial state distribution of the Markov chain induced by policy μ
q^μ	reward distribution of the Markov chain induced by policy μ

Notation	Definition
$q^{\mu}(\cdot\|s)$	reward distribution of the Markov chain induced by policy μ at state s
$R^{\mu}(s) \sim q^{\mu}(\cdot\|s)$	reward random variable of the Markov chain induced by policy μ at state s
$\pi^{\mu} \in \mathcal{P}(\mathcal{S})$	stationary distribution over states of the Markov chain induced by policy μ
$\mathcal{Z} = \mathcal{S} \times \mathcal{A}$	set of state-action pairs
$z = (s, a) \in \mathcal{Z}$	a nominal state-action pair
$\xi = \{z_0, z_1, \ldots, z_T\}$	a system path or trajectory
Ξ	set of all possible system trajectories
$\rho(\xi)$	(discounted) return of path ξ
$\bar{\rho}(\xi)$	expected value of the (discounted) return of path ξ
$\eta(\mu)$	expected return of policy μ
$D^{\mu}(s)$	random variable of the (discounted) return of state s
$D^{\mu}(z)$	random variable of the (discounted) return of state-action pair z
V^{μ}	value function of policy μ
Q^{μ}	action-value function of policy μ
V^{*}	optimal value function
Q^{*}	optimal action-value function
γ	discount factor
α	linear function over the belief simplex in POMDPs
Γ	the set of α-functions representing the POMDP value function
$k(\cdot, \cdot)$	a kernel function
$\boldsymbol{K}$	kernel matrix – covariance matrix – $[K]_{i,j} = k(x_i, x_j)$
$\boldsymbol{k}(x) = \big(k(x_1, x), \ldots, k(x_T, x)\big)^{\top}$	a kernel vector of size T
$\boldsymbol{I}$	identity matrix
$\mathcal{D}_T$	a set of training samples of size T
φ_i	a basis function (e.g., over states or state-action pairs)
$\boldsymbol{\phi}(\cdot) = \big(\varphi_1(\cdot), \ldots, \varphi_n(\cdot)\big)^{\top}$	a feature vector of size n
$\boldsymbol{\Phi} = [\boldsymbol{\phi}(x_1), \ldots, \boldsymbol{\phi}(x_T)]$	a $n \times T$ feature matrix
$\boldsymbol{\theta}$	vector of unknown parameters
Θ	a parameter space that the unknown parameters belong to ($\boldsymbol{\theta} \in \Theta$)

B

Discussion on GPTD Assumptions on the Noise Process

Assumption A2 *The residuals* $\Delta \boldsymbol{V}_{T+1} = \left(\Delta V(s_0), \ldots, \Delta V(s_T)\right)^\top$ *can be modeled as a Gaussian process.*

This may not seem like a correct assumption in general, however, in the absence of any prior information concerning the distribution of the residuals, it is the simplest assumption that can be made, because the Gaussian distribution has the highest entropy among all distributions with the same covariance.

Assumption A3 *Each of the residuals* $\Delta V(s_t)$ *is generated independently of all the others, i.e.,* $\mathbf{E}[\Delta V(s_i)\Delta V(s_j)] = 0$*, for* $i \neq j$*.*

This assumption is related to the well-known Monte-Carlo (MC) method for value function estimation [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998]. MC approach to policy evaluation reduces it into a supervised regression problem, in which the target values for the regression are samples of the discounted return. Suppose that the last non-terminal state in the current episode is s_{T-1}, then the MC training set is $\left\{\left(s_t, \sum_{i=t}^{T-1} \gamma^{i-t} r(s_i)\right)\right\}_{t=0}^{T-1}$. We may whiten the noise in the episodic GPTD model of Eqs. 5.9 and 5.10 by performing a

whitening transformation with the whitening matrix $\boldsymbol{H}^{-1}$. The transformed model is $\boldsymbol{H}^{-1}\boldsymbol{R}_T = \boldsymbol{V}_T + \boldsymbol{N}'_T$ with white Gaussian noise $\boldsymbol{N}'_T = \boldsymbol{H}^{-1}\boldsymbol{N}_T \sim \mathcal{N}\big(\boldsymbol{0}, \mathrm{diag}(\boldsymbol{\sigma}_T)\big)$, where $\boldsymbol{\sigma}_T = (\sigma_0^2, \ldots, \sigma_{T-1}^2)^\top$. The tth equation of this transformed model is

$$R(s_t) + \gamma R(s_{t+1}) + \ldots + \gamma^{T-1-t} R(s_{T-1}) = V(s_t) + N'(s_t),$$

where $N'(s_t) \sim \mathcal{N}(0, \sigma_t^2)$. This is exactly the generative model we would use if we wanted to learn the value function by performing GP regression using MC samples of the discounted return as our target (see §2.5.2). Assuming a constant noise variance σ^2, in the parametric case, the posterior moments are given by Eq. 2.22, and in the non-parametric setting, $\boldsymbol{\alpha}$ and $\boldsymbol{C}$, defining the posterior moments, are given by Eq. 2.18, with $\boldsymbol{y}_T = (y_0, \ldots, y_{T-1})^\top$; $y_t = \sum_{i=t}^{T-1} \gamma^{i-t} r(s_i)$. Note that here $\boldsymbol{y}_T = \boldsymbol{H}^{-1}\boldsymbol{r}_T$, where $\boldsymbol{r}_T$ is a realization of the random vector $\boldsymbol{R}_T$.

This equivalence uncovers the implicit assumption underlying MC value estimation that the samples of the discounted return used for regression are statistically independent. In a standard online RL scenario, this assumption is clearly incorrect as the samples of the discounted return are based on trajectories that partially overlap (e.g., for two consecutive states s_t and s_{t+1}, the respective trajectories only differ by a single state s_t). This may help explain the frequently observed advantage of TD methods using $\lambda < 1$ over the corresponding MC ($\lambda = 1$) methods. The major advantage of using the GPTD formulation is that it immediately allows us to derive exact updates for the parameters of the posterior moments of the value function, rather than waiting until the end of the episode. This point becomes more clear, when later in this section, we use the GPTD formulation and derive online algorithms for value function estimation.

References

Bbrl a c++ open-source library used to compare bayesian reinforcement learning algorithms. https://github.com/mcastron/BBRL/, 2015.

Y. Abbasi-Yadkori and C. Szepesvari. Bayesian optimal control of smoothly parameterized systems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2015.

P. Abbeel and A. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

S. Agrawal and N. Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Annual Conference on Learning Theory (COLT), JMLR W&CP*, volume 23, pages 39.1 – 39.26, 2012.

S. Agrawal and N. Goyal. Further optimal regret bounds for Thompson sampling. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, pages 99–107, 2013a.

S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 127–135, 2013b.

M. Araya-Lopez, V. Thomas, and O. Buffet. Near-optimal BRL using optimistic local transitions. In *International Conference on Machine Learning*, 2012.

J. Asmuth. *Model-based Bayesian Reinforcement Learning with Generalized Priors*. PhD thesis, Rutgers, 2013.

J. Asmuth and M. Littman. Approaching Bayes-optimality using Monte-Carlo tree search. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2011.

J. Asmuth, L. Li, M. Littman, A. Nouri, and D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2009.

K. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10: 174–205, 1965.

C. G. Atkeson and J. C. Santamaria. A comparison of direct and model-based reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 1997.

A. Atrash and J. Pineau. A Bayesian reinforcement learning approach for customizing human-robot interfaces. In *International Conference on Intelligent User Interfaces*, 2009.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

M. Babes, V. Marivate, K. Subramanian, and M. Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning*, pages 897–904, 2011.

A. Barto, R. Sutton, and C. Anderson. Neuron-like elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man and Cybernetics*, 13:835–846, 1983.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

J. Baxter, A. Tridgell, and L. Weaver. Knightcap: A chess program that learns by combining TD(λ) with game-tree search. In *Proceedings of the 15th International Conference on Machine Learning*, pages 28–36, 1998.

J. Baxter, P. Bartlett, and L. Weaver. Experiments with infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15: 351–381, 2001.

R. Bellman. *Dynamic Programming*. Princeton Universty Press, 1957.

D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

L. Bertuccelli, A. Wu, and J. How. Robust adaptive Markov decision processes: Planning with model uncertainty. *Control Systems, IEEE*, 32(5): 96–109, Oct 2012.

S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Incremental natural actor-Critic algorithms. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 105–112, 2007.

S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

J. Boyan. Least-squares temporal difference learning. In *Proceedings of the 16th International Conference on Machine Learning*, pages 49–56, 1999.

S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Journal of Machine Learning*, 22:33–57, 1996.

R. Brafman and M. Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 3:213–231, 2003.

S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012. ISSN 1935-8237.

R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

P. Castro and D. Precup. Using linear programming for Bayesian exploration in Markov decision processes. In *International Joint Conference on Artificial Intelligence*, pages 2437–2442, 2007.

P. Castro and D. Precup. Smarter sampling in model-based Bayesian reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, 2010.

M. Castronovo, D. Ernst, and R. Fonteneau A. Couetoux. Benchmarking for bayesian reinforcement learning. Working paper, Inst. Montefiore, http://hdl.handle.net/2268/185881, 2015.

G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A Bayesian approach. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013.

G. Chalkiadakis, E. Elkinda, E. Markakis, M. Polukarov, and N. Jennings. Cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research*, 39(1):179–216, 2010.

O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 2249–2257, 2011.

K. Chen and M. Bowling. Tractable objectives for robust policy optimization. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 2078–2086, 2012.

J. Choi and K. Kim. Map inference for Bayesian inverse reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1989–1997, 2011.

J. Choi and K. Kim. Nonparametric Bayesian inverse reinforcement learning for multiple reward functions. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 305–313, 2012.

R. Crites and A. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33:235–262, 1998.

P. Dallaire, C. Besse, S. Ross, and B. Chaib-draa. Bayesian reinforcement learning in continuous POMDPs with Gaussian processes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

R. Dearden, N. Friedman, and S. J. Russell. Bayesian Q-learning. In *AAAI Conference on Artificial Intelligence*, pages 761–768, 1998.

R. Dearden, N. Friedman, and D. Andre. Model based Bayesian exploration. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1999.

E. Delage and S. Mannor. Percentile optimization for Markov decision processes with parameter uncertainty. *Operations Research*, 58(1):203–213, 2010.

C. Dimitrakakis and C. Rothkopf. Bayesian multi-task inverse reinforcement learning. In *Proceedings of the European Workshop on Reinforcement Learning*, 2011.

F. Doshi, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: using Bayes risk for active learning in POMDPs. In *International Conference on Machine Learning*, 2008.

F. Doshi-Velez. The infinite partially observable Markov decision process. In *Proceedings of the Advances in Neural Information Processing Systems*, 2009.

F. Doshi-Velez, D. Wingate, N. Roy, and J. Tenenbaum. Nonparametric Bayesian policy priors for reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, 2010.

F. Doshi-Velez, J. Pineau, and N. Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. *Artificial Intelligence*, 2011.

M. Duff. Monte-Carlo algorithms for the improvement of finite-state stochastic controllers: Application to Bayes-adaptive Markov decision processes. In *International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2001.

M. Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts Amherst, Amherst, MA, 2002.

B. Efron. *Large-Scale Inference: Empirical Bayes Methods fro Estimation, Testing, and Prediction*. IMS Statistics Monographs. Cambridge University Press, 2010.

Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, The Hebrew University of Jerusalem, Israel, 2005.

Y. Engel, S. Mannor, and R. Meir. Sparse online greedy support vector regression. In *Proceedings of the 13th European Conference on Machine Learning*, pages 84–96, 2002.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 154–161, 2003.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 201–208, 2005a.

Y. Engel, P. Szabó, and D. Volkinshtein. Learning to control an octopus arm with gaussian process temporal difference methods. In *Proceedings of the Advances in Neural Information Processing Systems*, 2005b.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

A. Farahmand, M. Ghavamzadeh, C., and Shie Mannor. Regularized policy iteration. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 441–448, 2008a.

A. Farahmand, M. Ghavamzadeh, C. Szepesvári, and S. Mannor. Regularized fitted q-iteration: Application to planning. In *Recent Advances in Reinforcement Learning, 8th European Workshop, EWRL*, pages 55–68, 2008b.

M. M. Fard and J. Pineau. PAC-Bayesian model selection for reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*, 2010.

M. M. Fard, J. Pineau, and C. Szepesvari. PAC-Bayesian policy evaluation for reinforcement learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

A. Feldbaum. Dual control theory, parts i and ii. *Automation and Remote Control*, 21:874–880 and 1033–1039, 1961.

N. M. Filatov and H. Unbehauen. Survey of adaptive dual control methods. In *IEEE Control Theory and Applications*, volume 147, pages 118–128, 2000.

N. Friedman and Y. Singer. Efficient Bayesian parameter estimation in large discrete domains. In *Proceedings of the Advances in Neural Information Processing Systems*, 1999.

S. Gelly, L. Kocsis, M. Schoenauer, M. Sebag, D. Silver, C. Szepesvari, and O. Teytaud. The grand challenge of computer Go: Monte Carlo tree search and extensions. *Communications of the ACM*, 55(3):106–113, 2012.

M. Ghavamzadeh and Y. Engel. Bayesian policy gradient algorithms. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 457–464, 2006.

M. Ghavamzadeh and Y. Engel. Bayesian Actor-Critic algorithms. In *Proceedings of the 24th International Conference on Machine Learning*, pages 297–304, 2007.

M. Ghavamzadeh, Y. Engel, and M. Valko. Bayesian policy gradient and actor-critic algorithms. Technical Report 00776608, INRIA, 2013.

J. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

P. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33:75–84, 1990.

A. Gopalan and S. Mannor. Thompson sampling for learning parameterized markov decision processes. In *Proceedings of the 28th Conference on Learning Theory (COLT)*, pages 861–898, 2015.

A. Gopalan, S. Mannor, and Y. Mansour. Thompson sampling for complex online problems. In *Proceedings of the 31st International Conference on Machine Learning*, pages 100–108, 2014.

T. Graepel, J.Q. Candela, T. Borchert, and R. Herbrich. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In *Proceedings of the 27th International Conference on Machine Learning*, pages 13–20, 2010.

A. Greenfield and A. Brockwell. Adaptive control of nonlinear stochastic systems by particle filtering. In *International Conference on Control and Automation*, 2003.

A. Guez, D. Silver, and P. Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Proceedings of the Advances in Neural Information Processing Systems*, 2012.

S. Guha and K. Munagala. Stochastic regret minimization via Thompson sampling. In *Proceedings of The 27th Conference on Learning Theory*, pages 317–338, 2014.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the Advances in Neural Information Processing Systems*, 1999.

R. Jaulmes, J. Pineau, and J. Precup. Active learning in partially observable Markov decision processes. In *European Conference on Machine Learning*, 2005.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

E. Kaufmann, O. Cappé, and A. Garivier. On Bayesian upper confidence bounds for bandit problems. In *International Conference on Artificial Intelligence and Statistics*, pages 592–600, 2012a.

E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, volume 7568 of *Lecture Notes in Computer Science*, pages 199–213, 2012b.

K. Kawaguchi and M. Araya-Lopez. A greedy approximation of Bayesian reinforcement learning with probably optimistic transition model. In *Adaptive Learning Agents 2013 (a workshop of AAAMAS)*, 2013.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *International Conference on Machine Learning*, pages 260–268, 1998.

M. Kearns, Y. Mansour, and A. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *International Joint Conference on Artificial Intelligence*, pages 1324–1331, 1999.

J. Kober, D. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research (IJRR)*, 2013.

L. Kocsis and C. Szepesvari. Bandit based Monte-Carlo planning. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2006.

N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2619–2624, 2004.

J. Kolter and A. Ng. Near-Bayesian exploration in polynomial time. In *International Conference on Machine Learning*, 2009.

V. Konda and J. Tsitsiklis. Actor-Critic algorithms. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1008–1014, 2000.

M. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

A. Lazaric and M. Ghavamzadeh. Bayesian multi-task reinforcement learning. In *Proceedings of the 27th International Conference on Machine Learning*, pages 599–606, 2010.

L. Li. *A unifying framework for computational reinforcement learning theory*. PhD thesis, Rutgers, 2009.

C. Liu and L. Li. On the prior sensitivity of Thompson sampling. *CoRR*, abs/1506.03378, 2015.

S. Mannor and J. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *The Journal of Machine Learning Research*, 5:623–648, 2004.

S. Mannor, R. Rubinstein, and Y. Gat. The cross entropy method for fast policy search. In *International Conference on Machine Learning*, 2003.

S. Mannor, D. Simester, P. Sun, and J.N. Tsitsiklis. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007.

P. Marbach. *Simulated-Based Methods for Markov Decision Processes*. PhD thesis, Massachusetts Institute of Technology, 1998.

D. McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37, 1999.

N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289–312, 2008.

B. Michini and J. How. Bayesian nonparametric inverse reinforcement learning. In *Proceedings of the European Conference on Machine Learning*, 2012a.

B. Michini and J. How. Improving the efficiency of Bayesian inverse reinforcement learning. In *IEEE International Conference on Robotics and Automation*, pages 3651–3656, 2012b.

A. Moore and C. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

A. Neu and C. Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2007.

A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 663–670, 2000.

A. Ng, H. Kim, M. Jordan, and S. Sastry. Autonomous helicopter flight via reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*. MIT Press, 2004.

A. Nilim and L. El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

J. Niño-Mora. Computing a classic index for finite-horizon bandits. *INFORMS Journal on Computing*, 23(2):254–267, 2011.

A. O'Hagan. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29:245–260, 1991.

I. Osband, D. Russo, and B. Van Roy. (More) efficient reinforcement learning via posterior sampling. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 2013.

S. Paquet, L. Tobin, and B. Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 970–977, 2005.

J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: an anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003.

S Png. *Bayesian Reinforcement Learning for POMDP-based Dialogue Systems*. Master's thesis, McGill University, 2011.

S. Png and J. Pineau. Bayesian reinforcement learning for POMDP-based dialogue systems. In *ICASSP*, 2011.

H. Poincaré. *Calcul des Probabilités*. Georges Carré, Paris, 1896.

J. Porta, N. Vlassis, M. Spaan, and P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7, 2006.

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *International Conference on Machine learning*, pages 697–704, 2006.

W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality (2nd Edition)*. John Wiley & Sons, 2011.

M. Puterman. *Markov Decision Processes*. Wiley Interscience, 1994.

R. Munos R. Fonteneau, L. Busoniu. Optimistic planning for belief-augmented Markov Decision Processes. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2013.

D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2586–2591, 2007.

C. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 489–496. MIT Press, 2003.

C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems*. MIT Press, 2004.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

N. Ratliff, A. Bagnell, and M. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

R. Ravikanth, S. Meyn, and L. Brown. Bayesian adaptive control of time varying systems. In *IEEE Conference on Decision and Control*, 1992.

J. Reisinger, P. Stone, and R. Miikkulainen. Online kernel selection for Bayesian reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 816–823, 2008.

S. Ross and J. Pineau. Model-based Bayesian reinforcement learning in large structured domains. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2008.

S. Ross, B. Chaib-draa, and J. Pineau. Bayes-adaptive POMDPs. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 20, pages 1225–1232, 2008a.

S. Ross, B. Chaib-draa, and J. Pineau. Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *IEEE International Conference on Robotics and Automation*, 2008b.

S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 32:663–704, 2008c.

S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann. A Bayesian approach for learning and planning in partially observable Markov decision processes. *Journal of Machine Learning Research*, 12, 2011.

G. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University, 1994.

P. Rusmevichientong and J. N. Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411, 2010.

I. Rusnak. Optimal adaptive control of uncertain stochastic discrete linear systems. In *IEEE International Conference on Systems, Man and Cybernetics*, 1995.

S. Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 101–103, 1998.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2002.

D. Russo and B. Van Roy. An information-theoretic analysis of Thompson sampling. *CoRR*, abs/1403.5341, 2014a.

D. Russo and B. Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014b.

L. Scharf. *Statistical Signal Processing*. Addison-Wesley, 1991.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

S. Scott. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

Y. Seldin, P. Auer, F. Laviolette, J. Shawe-Taylor, and R. Ortner. PAC-Bayesian analysis of contextual bandits. In *Proceedings of the Advances in Neural Information Processing Systems*, 2011a.

Y. Seldin, N. Cesa-Bianchi, F. Laviolette, P. Auer, J. Shawe-Taylor, and J. Peters. PAC-Bayesian analysis of the exploration-exploitation trade-off. In *ICML Workshop on online trading of exploration and exploitation*, 2011b.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

R. Smallwood and E. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, Sep/Oct 1973.

V. Sorg, S. Sing, and R. Lewis. Variance-based rewards for approximate Bayesian reinforcement learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2010.

M. Spaan and N. Vlassis. Perseus: randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 24:195–220, 2005.

A. Strehl and M. Littman. A theoretical analysis of model-based interval estimation. In *International Conference on Machine learning*, pages 856–863, 2005.

A. Strehl and M. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74: 1209–1331, 2008.

M. Strens. A Bayesian framework for reinforcement learning. In *International Conference on Machine Learning*, 2000.

R. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

R. Sutton. DYNA, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2:160–163, 1991.

R. Sutton and A. Barto. *An Introduction to Reinforcement Learning*. MIT Press, 1998.

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

U. Syed and R. Schapire. A game-theoretic approach to apprenticeship learning. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1449–1456, 2008.

L. Tang, R. Rosales, A. Singh, and D. Agarwal. Automatic ad format selection via contextual bandits. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1587–1594. ACM, 2013.

G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6:215–219, 1994.

W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

J. N. Tsitsiklis. A short proof of the gittins index theorem. *The Annals of Applied Probability*, pages 194–199, 1994.

N. Vien, H. Yu, and T. Chung. Hessian matrix distribution for Bayesian policy gradient reinforcement learning. *Information Sciences*, 181(9):1671–1685, 2011.

T. Walsh, S. Goschin, and M. Littman. Integrating sample-based planning and model-based reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, 2010.

T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *International Conference on Machine learning*, pages 956–963, 2005.

C. Watkins. *Learning from Delayed Rewards*. PhD thesis, Kings College, Cambridge, England, 1989.

R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *Proceedings of the International Conference on Machine Learning*, pages 1015–1022, 2007.

B. Wittenmark. Adaptive dual control methods: An overview. In *5th IFAC symposium on Adaptive Systems in Control and Signal Processing*, 1995.

O. Zane. Discrete-time Bayesian adaptive control problems with complete information. In *IEEE Conference on Decision and Control*, 1992.

B. Ziebart, A. Maas, A. Bagnell, and A. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, pages 1433–1438, 2008.